Edith Cowan University

Research Online

Theses: Doctorates and Masters

Theses

1-1-1994

Applications of fuzzy counterpropagation neural networks to non-linear function approximation and background noise elimination

I. M. Wiryana
Edith Cowan University

Follow this and additional works at: https://ro.ecu.edu.au/theses

 Part of the Computer Sciences Commons

Recommended Citation

Wiryana, I. M. (1994). *Applications of fuzzy counterpropagation neural networks to non-linear function approximation and background noise elimination*. https://ro.ecu.edu.au/theses/1107

This Thesis is posted at Research Online.
https://ro.ecu.edu.au/theses/1107

Edith Cowan University

Copyright Warning

You may print or download ONE copy of this document for the purpose of your own research or study.

The University does not authorize you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site.

You are reminded of the following:

- Copyright owners are entitled to take legal action against persons who infringe their copyright.

- A reproduction of material that is protected by copyright may be a copyright infringement. Where the reproduction of such material is done without attribution of authorship, with false attribution of authorship or the authorship is treated in a derogatory manner, this may be a breach of the author's moral rights contained in Part IX of the Copyright Act 1968 (Cth).

- Courts have the power to impose a wide range of civil and criminal sanctions for infringement of copyright, infringement of moral rights and other offences under the Copyright Act 1968 (Cth). Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Applications of Fuzzy Counterpropagation Neural Networks to Non-linear Function Approximation and Background Noise Elimination

BY

I Made Wiryana

A Thesis submitted in Partial Fulfilment of
the Requirements for the Award of

Master of Applied Science (Computer Science)

at the School of Information Technology and Mathematics
Edith Cowan University

Date of Submission : 4 October 1994

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

Applications of Fuzzy Counterpropagation Neural Networks to Non-linear Function Approximation and Background Noise Elimination

Abstract

An adaptive filter which can operate in an unknown environment by performing a learning mechanism that is suitable for the speech enhancement process. This research develops a novel ANN model which incorporates the fuzzy set approach and which can perform a non-linear function approximation. The model is used as the basic structure of an adaptive filter. The learning capability of ANN is expected to be able to reduce the development time and cost of the designing adaptive filters based on fuzzy set approach. A combination of both techniques may result in a learnable system that can tackle the vagueness problem of a changing environment where the adaptive filter operates. This proposed model is called Fuzzy Counterpropagation Network (Fuzzy CPN). It has fast learning capability and self-growing structure. This model is applied to non-linear function approximation, chaotic time series prediction and background noise elimination.

Declaration

"I certify that this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any institution of higher education; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text"

Signature .

Date : 4 Ocotober 1994

Acknowledgements

I would like to express my sincere thanks to Dr. Hon Nien Cheung and Dr. Jim Millar for their help and supervision during the preparation and writing of this thesis.

I would like also like to express my gratitude to Australia Vice Chancellor Committee and STMIK Gunadarma who sponsor my study through ADCSS (Australia Development Co-operation Scholarship Scheme).

I am grateful to my family for their understanding, encouragement and support during these difficult times.

Thanks are also extended to the staff in the Department of Computer Science, and Department of Computer and Communication Engineering for their assistance and advice, and my friends and colleagues for their help and understanding while I was completing this thesis.

Table of Contents

		Page
Abstract		ii
Declaration		iv
Acknowledgements		v
Table of Contents		vi
List of Figures		viii
Principal Notations		xi
Chapter 1	Introduction	1
	1.1 Motivations	1
	1.2 Method of Investigations	5
	1.3 Outline of the Thesis	7
Chapter 2	Adaptive Filter	9
	2.1 Structure of Adaptive Filter	10
	2.1.1 Classification of adaptive system	11
	2.1.2 Time domain adaptive filter	14
	2.1.3 Frequency domain adaptive filter	17
	2.2 Adaptive Filter Algorithms	20
	2.2.1 Least Mean Square (LMS)	23
	2.2.2 Recursive Least Square (RLS)	25
	2.3 Application of Adaptive Filters for Signal Enhancement Problems	29
Chapter 3	Neural Network and Fuzzy System	35
	3.1 Artificial Neural Network	35
	3.1.1 Formal desription of a neuron	37
	3.1.2 Feature mapping in neural network	49
	3.1.3 Computation and learning in neural network	51
	3.2 Fuzzy System	58
	3.2.1 Fuzzy set	58
	3.2.2 Fuzzy system components	60
	3.3 Model-Free Function approximator	66
	3.3.1 Function approximation by neural network	67
	3.3.2 Function approximation by fuzzy system	71
Chapter 4	Fuzzy-CPN Model	76
	4.1 Proposed Model	76
	4.1.1 Formal description of Fuzzy-CPN	79
	4.1.2 Counterpropagation Paradigm	85
	4.1.3 Architecture of Fuzzy-CPN	90

	4.2	Fuzzy-CPN	95
		4.2.1 Adaptation mechanism	95
		4.2.2 Learning mechanism	100
		4.2.3 Recall mechanism	107
	4.3	Function Approximation in Adaptive Filter	110
		4.3.1 Fuzzy neuron as fuzzy rule	112
		4.3.2 Function approximation by Fuzzy-CPN model	115
Chapter 5	Non Linear Function Approximation		118
	5.1	Function Approximation Problem	118
	5.2	Comparison to another ANN Model	119
		5.2.1 Comparison to Counterpropagation Network (CPN)	123
		5.2.2 Comparison to the Interpolative CPN (ICPN)	126
	5.3	The Effect of Learning Parameter to the Performance	131
	5.4	Time Series Prediction	138
Chapter 6	Background Noise Elimination		144
	6.1	Noise Elimination	144
		6.1.1 Noise elimination model	145
		6.1.2 Background noise detection	148
		6.1.3 Time-frequency representation as pre-processing	150
		6.1.4 Spectral processing for the noise elimination	157
	6.2	Fuzzy-CPN for Background Noise Elimination	162
		6.2.1 Structure of the model	163
		6.2.2 Fuzzy-CPN as associative memory	169
	6.3	Simulations	171
Chapter 7	Conclusion		176
	7.1	Summary of Results	176
	7.2	Future Improvements	180
	7.3	Conclusions	181
References			183
Appendices			
	A	List of Acronyms	204
	B	Mackey-Glass Chaotic Time Series	207
	C	Example of Real Background Noise Elimination	209
	D	List of Equation	222
	E	Papers	231

List of Figures

		Page
Figure 2.1	Basic principle of adaptive filter	9
Figure 2.2	Adaptive system for signal identification	11
Figure 2.3	Adaptive System for signal estimation	12
Figure 2.4	Adaptive System for signal correction	13
Figure 2.5	Time domain adaptive filter	15
Figure 2.6	Linear Combiner Structure	16
Figure 2.7	Basic configuration of frequency domain filtering (Malvar, 1992)	18
Figure 3.1	A neuron structure (Hecht-Nielsen, 1987)	37
Figure 3.2	A formal neuron	38
Figure 3.3	Sigmoid function	41
Figure 3.4	Gaussian function	42
Figure 3.5	Process in neuron.	45
Figure 3.6	Neuron connection	46
Figure 3.7	Learning mechanism concepts	52
Figure 3.8	Membership function	59
Figure 3.9	Building block of fuzzy system	60
Figure 3.10	Gaussian membership function	62
Figure 3.11	Triangle membership function	62
Figure 3.12	Function approximation by fuzzy patch	71
Figure 3.13	Fuzzy patch as fuzzy rule (Kosko, 1993)	73
Figure 4.1	Fuzzy-neuron (FN)	80
Figure 4.2	The relation of the input space and output of the fuzzy neuron	82
Figure 4.3	Defuzzifier neuron (DN)	83
Figure 4.4	Counterpropagation mechanism	86
Figure 4.5	Fuzzy Counterpropagation Network architecture	89
Figure 4.6	Membership function	92
Figure 4.7	Learning mechanism in Fuzzy-CPN model	100
Figure 4.8	The receptive field adaptation of a fuzzy neuron	103
Figure 4.9	The shell distance S	106
Figure 4.10	Training result of a 2-dimensional input space	107
Figure 4.11	Recall mechanism	108
Figure 4.12	Mechanism of adaptive system	109
Figure 4.13	Fuzzy neuron as fuzzy rule	113
Figure 5.1	Function approximation	119
Figure 5.2	The function of Eq.5.1.	120
Figure 5.3	The Fuzzy CPN used	121
Figure 5.4	Approximation Eq.5.1. using Fuzzy CPN	122
Figure 5.5	The CPN used	123
Figure 5.6	Approximation CPN to the Eq.5.1.	124
Figure 5.7	The diference approach between CPN and Fuzzy CPN	125
Figure 5.8	Approximation by the ICPN to the Eq.5.1.	128
Figure 5.9	The difference approach between ICPN and FCPN approach.	129
Figure 5.10	Comparison of CPN, ICPN and Fuzzy-CPN (FCPN)	131
Figure 5.11	Number of neuron generated versus ρ	133

Page

Figure 5.12	MSE versus ρ	134
Figure 5.13	MSE versus number of neuron percentage	135
Figure 5.14	MSE versus α_{limi}	137
Figure 5.15	Mackey-Glass chaotic time series	139
Figure 5.16	Simulation using x_{500} to x_{700} as training data	140
Figure 5.17	Simulation using x_0 - x_{700} as training data	141
Figure 5.18	Simulation using on-line adaptation x_{500} - x_{700} as training pattern	142
Figure 5.19	Simulation using on-line adaptation x_0 - x_{700} as training pattern	142
Figure 6.1	General model of noise elimination	146
Figure 6.2	Signal $s_1(t)$ in Eq.6.9 with $p = 56$, $l = 300$	153
Figure 6.3	Signal $s_2(t)$ in Eq.6.9. with $p = 256$, $l = 300$	153
Figure 6.4	Spectrogram of $s_1(t)$	154
Figure 6.5	Spectrogram of $s_2(t)$	154
Figure 6.6	Contour diagram of Figure 6.4	154
Figure 6.7	Contour diagram of Figure 6.5	155
Figure 6.8	Proposed spectrogram of $s_1(t)$	155
Figure 6.9	Proposed spectrogram of $s_2(t)$	155
Figure 6.10	Contour diagram of Figure 6.8	156
Figure 6.11	Contour diagram of Figure 6.9	156
Figure 6.12	Speech enhancement process using STFT (Boll, 1987)	159
Figure 6.13	Input signal acquisition process.	165
Figure 6.14	Frames of the signal	168
Figure 6.15	Fuzzy-CPN (FCPN) as associative memory	169
Figure 6.16	Sinusoidal input signal is corrupted by gaussian noise with SNR equal to 17.49 dB	172
Figure 6.17	Output signal after filtering, with SNR equal to 27.07 dB	172
Figure 6.18	Sinusoidal input signal with amplitude modulated, with SNR equal to 20.31 dB	173
Figure 6.19	Output signal after filtering, with SNR equal to 25.52 dB	173
Figure 6.20	Sinusoidal input signal with frequency modulated, with SNR equal to 12.37 dB	174
Figure 6.21	Output signal after filtering with SNR equal to 29.03 dB	174
Figure B.1	Mackey-Glass chaotic time series	208
Figure C.1	Time-domain representation of street noise	210
Figure C.2	Conventional spectrogram of signal shown in Figure C.1	211
Figure C.3	Contour diagram of Figure C.2	211
Figure C.4	Proposed spectrogram of signal shown in Figure C.1	212
Figure C.5	Contour diagram of Figure C.4	212
Figure C.6	Time-domain representation of foot step noise	213
Figure C.7	Conventional spectrogram of signal shown in Figure C.6	214
Figure C.8	Contour diagram of Figure C.7	214
Figure C.9	Proposed spectrogram of signal shown in Figure C.6	215
Figure C.10	Contour diagram of Figure C.9	215
Figure C.11	Time-domain representation of crowded noise	216
Figure C.12	Conventional spectrogram of signal shown in Figure C.11	217
Figure C.13	Contour diagram of Figure C.12	217
Figure C.14	Proposed spectrogram of signal shown in Figure C.11	218

Page

Figure C.15 Contour diagram of Figure C.14 218
Figure C.16 Input signal : sinousidal and street noise 219
Figure C.17 Output signal 219
Figure C.18 Input signal : sinusoidal with crowd cheering noise 220
Figure C.19 Output signal 220
Figure C.20 Input signal : sinusoidal with troop marching noise 221
Figure C.21 Output signal 221

Principal Notations

In this thesis following notations are used :

- A scalar is denoted using normal font, italic, and lowercase

  e.g. : x, y, z etc.

- A vector is denoted using bold typeface, non-italic and in lowercase.

  e.g. : $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ etc.

  Component of a vector is denoted as scalar with single index

  e.g. : $\mathbf{w} : [w_1, w_2, \ldots, w_N]$

- A matrix is notated using bold typeface, non-italic and in uppercase.

  e.g. : $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{Z}$ etc.

  Components of a matrix are denoted as scalar with index more than one.

  e.g : $\mathbf{X} = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \end{bmatrix}$

- A set is denoted as

  $X = \{x_1, x_2, \ldots, x_n\}$

- An ordered set is denoted as

  $X = [x_1, x_2, \ldots, x_n]$

- A function is notated using italic

  e.g.: $f(\mathbf{x} \mid \mathbf{w})$

  where f is the function

  $\mathbf{x}$ is the set of function variable

  $\mathbf{w}$ is the set of function parameters

- A mapping function is notated as :

  $f: R^N \rightarrow R^M$

  f is a mapping function from the domain with N-dimensional space to the range with M-dimensional space.

Chapter 1
Introduction

1.1 Motivations

Speech transmission and processing are often degraded by acoustic or electrical noises. In order to reduce and eliminate this degradation effect in speech transmission, a speech enhancement process is employed. This process improves the quality and intelligibility of the received signal by pre-processing or post-processing. Various methods have been proposed to perform speech enhancement, including fixed filtering and adaptive filtering. An adaptive filter is used when there is not enough information about the desired signal and the environment.

An adaptive system consists of a time-varying digital signal processing system that learns to perform a particular transformation with respect to the input signal to be able to adapt to the environment by using an iterative process. The task of adaptive filters can be classified as system identification, signal estimation and signal correction.

An adaptive filter uses some quality criteria to perform the adaptation either in the time domain or in the frequency domain. Most of the conventional adaptive algorithms use the mean square error as the quality criterion, such as Least Mean Square (LMS) algorithm and Recursive Least Square (RLS) algorithm. LMS is the most popular but has some limitations: the initial convergence is slow, the convergence of the system depends on the input signal characteristics and a residual error still exists after convergence. Although, RLS has better performance than LMS algorithm, this algorithm requires an initialisation matrix, and for some problems the

execution of the algorithm becomes numerically unstable and impractical due to computational resource limitations.

The conventional adaptive filter approaches typically employ some form of linear adaptive filters. They have proved valuable in signal processing problems where training data are available. However, for non-linear problems, a solution is difficult to obtain by using the conventional linear adaptive filter. Non-linear techniques have been developed as well, but these typically depend on a local linearisation of the problem. The failure of conventional adaptive filters to solve the non-linear problems leads to the development of an adaptive filter using a non-linear processing model such as an Artificial Neural Network (ANN) or a Fuzzy System (FS).

By employing the ANN techniques, less assumptions are required to build a non-linear adaptive filter model, because the ANN has capability to learn the input-output relation of the non linear relationship. The ANN model is an alternative form of information processing that is a fundamentally new and different information processing paradigm. ANN models have been considered particularly suitable for unstructured computations. They have been proved to be more robust when the signal distributions are generated by a non-linear processes and are strongly non-Gaussian. Basically, an ANN system consists of a number of interconnected neurons and most of them are arranged in the form of multi layer structure. There is the natural concept of a sequential flow of information which is a feature mapping from each layer to the subsequent upper layer.

There are two phases of operation for an ANN, the learning phase and the recall phase. During the learning phase, by employing a learning scheme, an ANN

constructs the hypersurface function from sparse training data point by optimising the cost function in order to reach a global minimum. The result of the training yields a network which simulates the implemented function as closely as possible. This implemented function is used by the ANN to produce the output during the recall phase.

Uncertainty of the input signal and the environment have been problems in the designing of an adaptive system. To deal with uncertainty, several paradigms such as probability or possibility may be used. Fuzzy systems which depend on possibility theory is one of the approaches in dealing with ambiguity or vagueness.

Both ANN and FS techniques are model-free estimators that can estimate a function without knowing the mathematical model of the input-output relation. As stated by the existence theorem from Kolmogorov, an ANN model can implement a continuous function (Hecht-Nielsen, 1987). In the case of FS technique, a fuzzy system (FS) can be used as a universal approximator which is ensured by Stone-Weierstrass theorem (Wang, 1992).

This research intends to develop a novel ANN model which incorporates the fuzzy approach that can be used as the basic block for adaptive filter. The learning capability of ANN is expected to be able to reduce the development time and cost of designing the fuzzy system A combination of both techniques may produce a learnable system that can tackle the vagueness problem.

To build a basic structure of an adaptive filter that incorporate the ANN model with fuzzy approach, some application constraints should be considered:

- there is no clean signal as the target signal for training purposes;
- learning must be performed with a small number of iterations;

- the neural network should be able to adapt its structure to minimise memory usage;
- fast processing time in the recall phase is required for real-time operation.

The proposed model is called **Fuzzy-CPN**, using the Counterpropagation (CPN) paradigm by incorporating the fuzzy set approach at the middle layer and by implementing the fuzzy leader clustering technique. This model can be trained faster and has a self-growing structure.

Basically, the Fuzzy-CPN model performs the function approximation of any arbitrary function that can be epresented by a training set. Some parameters of the model affect the performance of the model in approximating a function. Study of the influence of the learning parameter on the overall performance is crucial before applying the model to the adaptive filter task.

Signal estimation is one class of the adaptive filter tasks that can be solved by using this proposed model. The Mackey-Glass chaotic time series prediction is performed by making use of the function approximation capability of this model. By employing the learning mechanism of the Fuzzy-CPN using the input-output pair of the past data and the present data, a prediction of the future data is obtained.

The proposed model will also be applied to tackle the signal correction in a background noise elimination system which makes use of time-frequency representation of the input signal and a spectral subtraction filtering process. The elimination of background noise in applications where an uncorrupted input signal is required is not a trivial task, especially when the noise is non-deterministic and non-stationary. In addition, the duration of the noise may be short compared with the observation intervals for the input signal. Background noise is an example of a

non-deteministic process that has no particular deterministic structure. In the application of background noise elimination in speech acquisition in real environment, the speech signal depends on the speaker, on the environmental conditions and on the transmission characteristics. Therefore, speech is a non-stationary and non-deterministic signal as well. In speech processing, the noise elimination process is to detect and eliminate the three noise components namely, the deterministic and stationary internal noise, the non-deterministic and stationary noise, and more importantly the non-deterministic and non-stationary noise, from the non-stationary speech source.

1.2 Methods of Investigation

This thesis intends to address the problems associated with adaptive signal filtering in signal processing, especially the signal enhancement problem. Instead of using the conventional predictive method, the model that will be developed makes use of the techniques in ANN that incorporate the fuzzy set approach with a modified learning method. The system will be able to deal with complex, non-stationary, uncorrelated and non-deterministic signals by applying the proposed model.

The significance of the implementation in using the ANN model proposed in this thesis will improve the potential of ANN in digital signal processing systems, especially speech processing applications. It is perceived that the use of such a model could contribute to an increase in the performance of adaptive digital filter systems in speech processing systems.

The first stage of this research is the investigation of a global definition of speech enhancement problems associated with the use of ANN. This research stage

also provides information on the issues in speech enhancement in speech processing. Furthermore, the available conventional techniques are reviewed in this stage.

The second stage is theoretical research which will be used in the development of a model for a signal enhancement system using an ANN as the adaptive filter. In this stage, a novel model is developed by taking some considerations. The first consideration is the speed of processing. Although ANNs are inherently parallel, but it requires a special hardware to implement them in parallel due to the requirement of complex intercommunication between simple processing units. The second consideration is the mechanism of performing the training of ANN. It may be on-line or off-line learning. On-line learning is desirable because the network would potentially be able to adjust to changes in the system. However, due to speed considerations, it may be necessary to train the network off-line and use it as a non-adaptive system.

The development of the proposed model defines the specifications of the model. Firstly, the configuration of the network is specified. The specification includes architecture, topology, number of layers, number of nodes, type of non-linearity, and associated parameters. They determine the computational characteristics and the power of the neural network. Secondly, the teacher or the target pattern for training phase is to be decided. When supervised learning is applied, the type of teacher which is available to give the correct output has to be chosen. Thirdly, the training set must be specified to give an adequate representation of the type of inputs. Finally an appropriate learning algorithm has to be chosen. A more efficient learning algorithm must be used to speed training and to guarantee the convergence without degrading the performance of the model.

The third stage of this research is applying the model to a problem that requires an adaptive filter. The system is targeted to use a speech signal as input and to produce a processed speech signal as output.

1.3 Outline of the Thesis

This thesis is organised as follow :

Chapter 1 is the Introduction. Followed by the general literature review on the theoretical framework in Chapter 2, the conventional linear adaptive filter is described. In this chapter, the discussion covers the basic structure of adaptive filters and their classifications. The conventional adaptive algorithms, Least Mean Square (LMS) and Recursive Least Square (RLS), briefly are described. The role of adaptive filter in the speech processing problem is also stated in this chapter.

Chapter 3 is gives a detail description of the Artificial Neural Network and Fuzzy System which will be used as the basic model of the proposed model in this work. Definitions of the terms, the formal neuron, and the architecture of an Artificial Neural Network (ANN) are covered in this chapter.

Chapter 4 covers the development of the Fuzzy-CPN including the definitions of the basic processing units and the connections between them, as well as the learning and recall mechanism of this structure. Since this architecture has a self-growing structure, the adaptation mechanism of the structure is described as well.

Chapter 5 and 6 detail the application of the proposed model in solving non-linearity problem. In Chapter 5, the capability of the Fuzzy-CPN in non-linear function approximation is shown. The influence of the learning parameters on the

network performance is described. An application of this proposed model to the prediction of the Chaotic Time Series problem is described.

In Chapter 6, the capability of the Fuzzy-CPN to perform associative memory is described. This capability is applied to build a background noise elimination system by combining the Fuzzy-CPN with several digital signal processing techniques including noise power estimation, time-frequency representation of the signals, and the spectral subtraction method for the filtering process.

Finally, in Chapter 7, conclusions of this work and the further improvements of the proposed model are described.

Chapter 2

Adaptive Filter

An adaptive system consists of a time-varying digital signal processing system that learns to perform a particular transformation with respect to the input signal. It is able to adapt to the environment, by using an iterative process. The basic structure of an adaptive filter is shown in Figure 2.1. An adaptive filter consists of an adjustable digital filter, with variable and adaptive coefficients, and a coefficient updating algorithm to adjust the variable filter coefficients on the basis of a measurement of the actual performance, e.g. mean square error, at regular intervals.

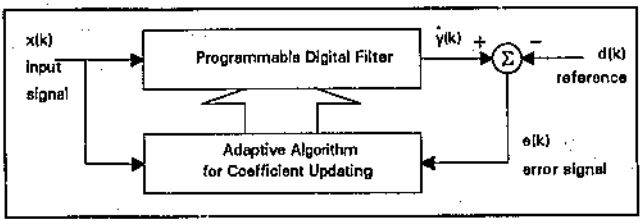

Figure 2.1 Basic principle of adaptive filter

As shown in Figure 2.1, after the input signal of the adaptive filter $x(k)$ is fed into the system, the programmable digital filter produces the a priori output $\hat{y}(k)$ by performing the filter function with the current coefficient values. There is an error signal $e(k)$ produced after comparing $\hat{y}(k)$ to the desired signal $d(k)$. The coefficients of the programmable digital filter are adjusted in order to minimise a pre-defined cost function with respect to the filter coefficients. The mean square error between filter output $\hat{y}(k)$ and an appropriate reference signal $d(k)$ is a typical cost function (Matthews, 1990). Using the error signal $e(k)$ and the input $x(k)$, the adaptive

algorithm minimises $e(k)$ until an optimal value is reached. The adaptive system will continually adjust its coefficients to reduce the error signal between the desired signal and its output. In this way, the system performs adaptation to its prescribed signal environment. Therefore, for performing coefficient adjustment, the adaptive digital filter needs a desired response and a performance criterion (Stearn, 1988).

2.1 Structure of Adaptive Filter

An adaptive system has some fundamental characteristics. Firstly, the task of an adaptive system will be influenced by an unknown system. Secondly, an adaptive system will only be able to start the filtering process after the unknown system becomes operative. The filtering process is established when a set of input signal has been fed into the system after an interval time. Finally, an adaptive system has to learn the function which has to be performed. The decision will be made according to measurement of the input signal. Therefore, it takes some time before a sufficiently reliable result is produced. As a result, in building an adaptive system, the basic system which is an adjustable digital filter must be able to change its parameters. Since the adaptive system works according to a quality criterion assessment, a quality criterion must be available. This criterion depends on the purpose of the system. In order to vary the coefficients of the adjustable digital filter, an algorithm must be found that is able to estimate the values of the coefficients.

An adaptive system can be classified according to the following features: the quality criterion used to optimise the system, the algorithm used to determine the coefficients, and the signal processing device used to build the system. Changing one of these features will yield a different class of adaptive systems.

2.1.1 Classification of adaptive systems

Adaptive systems can be divided to three major classes :

System Identification

This class of adaptive systems is shown in the Figure 2.2.

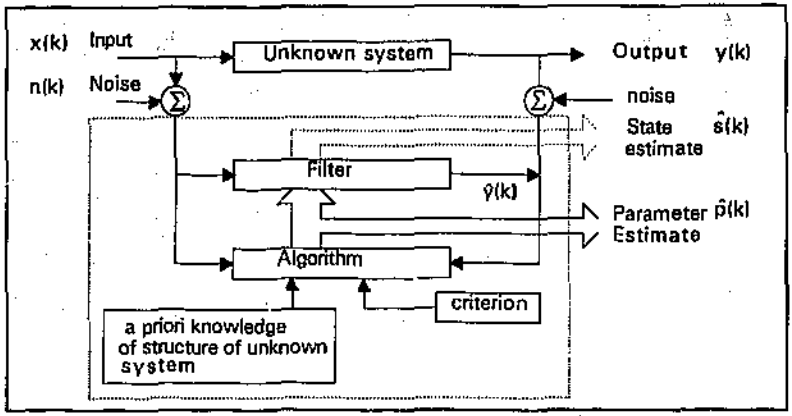

Figure 2.2 Adaptive system for signal identification

In Figure 2.2, the adaptive system takes measurements $x(k)$ and estimates the numerical values of the parameters $\hat{p}(k)$ or the state of the unknown system under study $\hat{s}(k)$, at a certain instant. These real or vector values produced by the estimation of the adaptive system describes the unknown system. For this purpose, a priori knowledge about the system is needed to access to both the input and output signals of that system. This problem is similar to the combination of parameter estimation and hypothesis testing in the statistical signal processing (Scharf, 1991). The adaptive filter system in this class adjusts its coefficients to produce a response that is as close as possible to the unknown system's response. If the internal noise of

the unknown system is small, the adaptive system will adapt to become a good model of the unknown system.

In the situation where the output of the unknown system $y(k)$, is used as the desired signal, the system is known as the forward modelling system, and it has a wide range of applications in biological, social, and economic science (Widrow and Stearn, 1985), digital filter design (Mutluay and Fahmy, 1984), geophysics (Magotra et al., 1991). When the input signal, which is passed through a delay processing, is used as the desired signal, the model is classified as the inverse modelling. This kind adaptive systems adjust the coefficients in order to become the inverse of the unknown system. The inverse modelling has been applied in many application such as channel equalisation (Qureshi, 1985).

Signal Estimation

The goal of this class is to estimate the output signal of an unknown system. It is used when the input signal is distorted. Typical examples of this class are echo canceller and noise suppresser (Gerald et al., 1990). This class is shown in Figure 2.3.

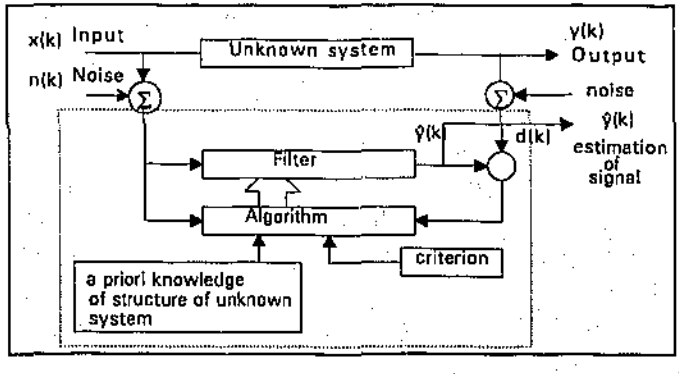

Figure 2.3 Adaptive system for signal estimation

As shown in Figure 2.3 the adaptive system of this class takes measurement of the input signal $x(k)$ and then estimates the signal $\hat{y}(k)$. The output signal of the unknown system is used as the desired signal $d(k)$ for the adaptive system. The adaptive system which will minimise an error between the actual output signal $y(k)$ and estimated output signal $\hat{y}(k)$. The system makes the output as close as possible to the future value of the input. The applications of this model are found in speech encoding (Horvarth, 1983), spectral estimation (Vary, 1983), event detection (Magotra et al., 1991), line enhancement (Soleit, 1988), and other areas.

Signal Correction

This system is applied when some unknown source has altered the characteristic of the signal. To be able to perform correction, a well defined criterion of signal quality must be available. This class of adaptive system is shown in Figure 2.4.

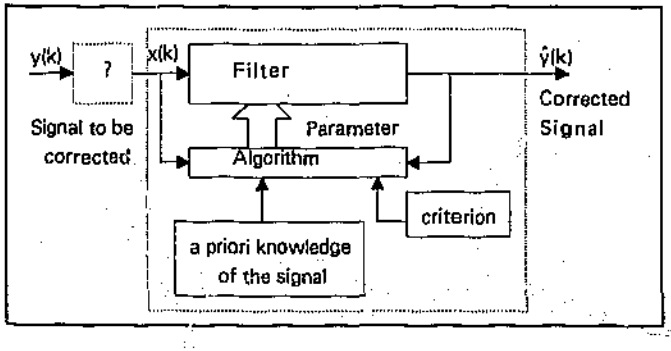

Figure 2.4 Adaptive system for signal correction

In Figure 2.4, an original input signal $y(k)$ passes through the unknown system. The signal is distorted by the system. The unknown system produces $x(k)$ which is the distorted version of $y(k)$. The filter estimates the signal $x(k)$ and produces the $\hat{y}(k)$ as

the estimated output in order to produce the original signal $y(k)$. Since the desired signal is not used in the adaptive processing, several techniques including prediction, filtering, and smoothing are used to extract information about the future, present and past from an input signal that is known to have some regularity or smoothness.

The applications of this model can be found in adaptive interference cancelling (Treichler et al., 1986), where the correlated noise is used as the criterion, and the input signal is the signal corrupted by the noise. The system will try to minimise the error by making the output adaptive subsystem approximate the noise. Therefore, it can be used to eliminate the noise in the signal.

The parameters of the system are updated according to a quality criterion. Decision of the criterion used is an important issue in this class of adaptive filters. There are two major methods: Exact Least Square and Gradient Method. The Exact Least Square performs the calculation of the mean square error after the system has collected a block of data. Consequently, it is slow to track the change of environment. The Gradient method estimates the error when each sample arrives and it offers a faster solution for tracking the change of environment (Classen et al., 1983).

2.1.2 Time domain adaptive filter

The simplest way in implementing adaptive filter algorithm is time delay tap structure. This filter structure is depicted in Figure 2.5, where z^{-1} is a time delay unit and $x(k)$ is the input signal at time index k.

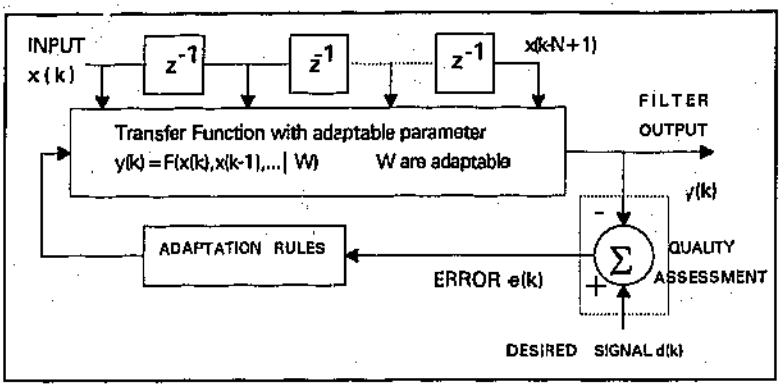

Figure 2.5 Time domain adaptive filter

The input signal $x(k)$ and previous input signal [$x(k\text{-}1)$, $x(k\text{-}2)$ ..., $x(k\text{-}N\text{+}1)$] are used together with a function having adaptable parameters $F(\ldots|\mathbf{w})$ is used to produce the output signal $y(k)$. After the function produces the output signal $y(k)$, an adaptation rule is applied to change the parameters w in the function $F(\ldots|\mathbf{w})$. This adaptation is performed in order to reduce the error $e(k\text{+}1)$ when a new input signal $x(k\text{+}1)$ arrives.

The most widely used function $F(\ldots.|\mathbf{w})$ is a line combiner. The output of this function is a linear combination of some weights $w(k)$ and the inputs $x(k)$. In general, can be written as :

$$y(n) = \sum_{k=-M}^{M} w(k)x(n-k) \qquad (2.1)$$

If M is ∞ the filter is known as Infinite Impulse Response Filter (IIR), otherwise it is know as Finite Impulse Response Filter (FIR). The structure of which is shown in Figure 2.6.

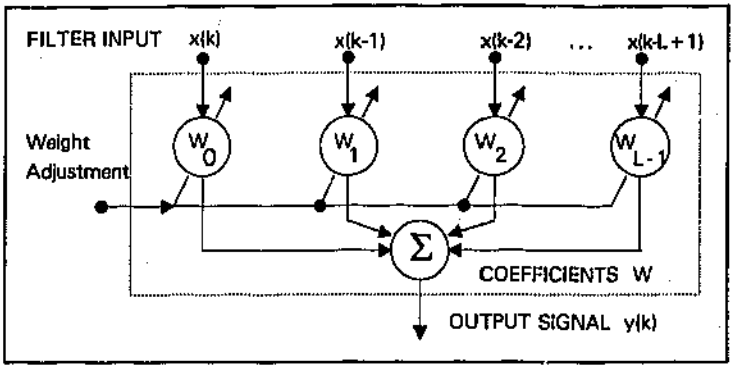

Figure 2.6 Linear combiner structure

The output signal $y(k)$ of the linear combiner is the linear combination of the input signal and the adjustable weights. The adaptive algorithm will adjust the weight values $\{w_0, w_1, \ldots, w_{L-1}\}$ according to a performance measurement. The advantage of this method is that it is easy to implement. However, this is a linear system and therefore it cannot be used for any non-linear tasks, such as non-linear adaptive equalisation.

The complexity of the algorithm is dictated by the number of taps, L, which determines the delay of the filter, i.e. the delay will be longer with increasing the number of taps. In a typical system, the impulse response is very long. Therefore, in order to satisfy the impulse response of this system, an FIR with many filter coefficients should be used. On the other hand, an IIR structure can be built using a recursive structure so that a long impulse response filter can be implemented only using a few filter coefficients. The complexity of IIR model is lower than that of FIR. One of the disadvantages of the IIR structure is that the poles of the filter can move outside the unit circle in the z-plane during adaptation, and hence it may lead to instability, slow convergence and local minima (Shynk, 1992).

The proposed model can be used in a time-domain adaptive filter structure and it replaces the linear combiner structure with a non-linear processing model, which will be described in the Chapter 5, by applying the proposed model to tackle time series prediction. This task is an example of adaptive filters used in signal estimation.

2.1.3 Frequency domain adaptive filter

In this structure, a frequency transformation is performed i.e., Fourier Transformation, before the adaptive algorithm is applied. The adaptation and the filtering are performed in the frequency domain, rather than in the time domain. The basic operation of a frequency-domain adaptive filter begins first by transforming the input signal into a frequency representation. It is achieved by an orthogonal transforms, such as Discrete Fourier Transformation (DFT) or Discrete Cosine Transformation (Malvar, 1992), of the input signal into frequency-domain. The desired signal, and the error signal, which is the difference between the output signal and the desired signal, are transformed into frequency domain as well. By performing an adaptive algorithm, the weights of the filter in the frequency domain will be adjusted to produce an output signal which approximates the desired signal. The adaptive algorithm in frequency domain is a version of the complex adaptive algorithm (Shynk, 1992). An adaptive filter can be implemented on a block by block basis using the Fast Fourier Transform (FFT) and is shown in following diagram :

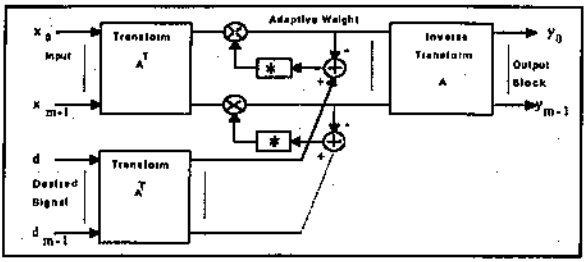

Figure 2.7 Basic configuration of frequency domain filtering (Malvar, 1992)

Suppose F is the frequency transformation function.

$$x(k) \overset{F}{\leftrightarrow} X(e^{j\omega}) \tag{2.2}$$

$$T(e^{j\omega}) = \Psi(X(e^{j\omega}), Y(e^{j\omega})) \tag{2.3}$$

Where $x(k)$ is the input signal and $y(k)$ is the output signal in the time domain representation. whereas $X(.)$, is the input signal and $Y(.)$ is the output signal in the frequency domain representation. $\Psi(.)$ is a spectral manipulation function, i.e. spectral subtraction or spectral multiplication. T (.) is the frequency domain representation of the desired signal. The following equation shows the connection between the frequency domain and time domain operation.

$$x(k) \otimes y(k) \overset{F}{\leftrightarrow} X(e^{j\omega}) \cdot Y(e^{j\omega}) \tag{2.4}$$

In Eq 2.4 $\otimes$ is the circular convolution. According the Eq. 2.4, the multiplication in the frequency domain is equivalent to the circular convolution in time domain (Oppenheimer, 1989). Therefore, to achieve a linear convolution for filtering purpose by implementing a DFT algorithm, there are two approaches, the overlap-save and overlap-add methods. By overlapping element of the input data and retaining only a part of the final DFT products, a linear convolution is obtained.

The frequency domain adaptive filtering method has primarily three advantages. Firstly, adaptive filtering using a block by block processing approach which is based on a fast transform could significantly reduce the computational complexity. An orthogonal transformation, i.e. FFT, can perform a linear convolution more efficiently than the conventional linear combiner structure. Secondly, the DFT structure can generate signals that are approximately uncorrelated. Thirdly, the frequency-domain adaptive filter converges faster (Shynk, 1992), than the time domain adaptive filter.

Since the adaptation and the filtering are performed in a block by block manner, this structure yields problems in the block boundaries. It leads to small discontinuities in the output signal $y(k)$. In applications where the input signal is already noisy, this lack of shift invariance is a minor concern. The boundary effect can be reduced by applying Orthogonal Lapped Transformation, but the complexity of the calculation will increase (Malvar, 1992). This approach may introduce longer end-to-end delay, because adaptation that the output filter produces after one block has been accumulated. For non-stationary signals, the tracking performance of a block algorithm also generally becomes worse especially for large and highly non-stationary input data (Shynk, 1992).

Another approach of the frequency domain adaptive filters is described in Narayan (Narayan, 1983). In this structure, there is no inverse transformation required. This filter structure is a modification of the transversal filter in which the output of the tapped delay line is first transformed before being combined by the adaptive weights. The orthogonal transform acts as a filter banks, so that its output is uncorrelated. The convergence of this filter is faster than a LMS transversal filter.

In this approach, the complexity is reduced because the algorithm does not perform the Inverse Transformation, and the speed of tracking is better because the transformation is performed by sliding the window (Narayan, 1983).

In this work the function $\Psi(.)$ is performed by the associative memory mechanism of the proposed model. It is expected to be able to enrich the system to deal with the non linear problem. It will be addressed in Chapter 6 for the noise elimination problem which is one of the applications of signal correction.

2.2 Adaptive Filter Algorithms

There are many adaptive algorithms which have been available. Most of them use the mean square error as the quality criterion (Classen, 1983). These algorithms are Exact Least Square, Least Mean Square, Recursive Least Square, Fast Least Square, and the variations of these algorithms which optimise the computation process to reduce the complexity of algorithms.

It is assumed that all of the signals are stationary and have finite correlation function. In order to describe the algorithms, some notations are defined as follows:

The input vector is :

$$\mathbf{x}_k = [x_k, x_{k-1}, \ldots, x_{k-L+1}]^T \tag{2.5}$$

L is the number of tap delay.

The coefficient vector is :

$$\mathbf{w} = [w_0, w_1, \ldots, w_{L-1}]^T \tag{2.6}$$

Let $d(k)$ is the desired signal and $y(k)$ is the output signal at time k. Then the error signal can be represented as :

$$e(k) = d(k) - y(k) \tag{2.7}$$

The crosscorelation between input signal $x(k)$ and output $d(k)$ is

$$r_{dx}(n) = E[d_x x_{k+n}] \tag{2.8}$$

The autocorrelation between input signal $x(k)$ and $x(k+n)$ is :

$$r_{xx}(n) = E[x_k x_{k+n}] \tag{2.9}$$

The crosscorrelation matrix **R** is given by :

$$\mathbf{R} = E\left[\mathbf{x}_k \mathbf{x}_k^T\right] \tag{2.10}$$

This **R** matrix is L x L size, symmetric, and a Toeplitz matrix. Therefore it is easy to invert if not singular:

$$R = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \dots & r_{xx}(L-1) \\ r_{xx}(1) & r_{xx}(0) & \dots & r_{xx}(L-2) \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ r_{xx}(L-1) & r_{xx}(L-2) & \dots & r_{xx}(0) \end{bmatrix} \tag{2.11}$$

The crosscorelation vector is:

$$\mathrm{P} = E[d_x \mathbf{x}_x] = E[\mathbf{x}_k d_k] \tag{2.12}$$

Since the coefficients are fixed for the moment, the output signal $y(k)$ is stationary, and the MSE is given by:

$$\begin{aligned} MSE &= E\left[e_k^2\right] = E\left[(d_k - y_k)^2\right] \\ &= E\left[d_k^2\right] + E\left[y_k^2\right] - 2E[d_k y_k] \\ &= r_{dd}(0) + r_{yy}(0) - 2r_{dy}(0) \end{aligned} \tag{2.13}$$

As shown by Stearns (Stearns, 1988), the MSE of an Adaptive Line Combiner such as a Finite Impulse Response Filter (FIR) which follows Eq. 2.1 can be written as :

$$MSE = r_{dd}(0) + \sum_{i=0}^{L-1}\sum_{i=0}^{L-1} w_i w_m r_{xx}(i-m) - 2\sum_{i=0}^{L-1} w_i r_{xd}(i) \qquad (2.14)$$

where w is the adjustable weight vector but for the moment is fixed. The MSE surface is an L-1 dimensional surface in the L dimensional space, where L is number of adjustable weights. The adaptation process is to seek the minimum point on this error surface.

The MSE can in Eq. 2.14 be rewritten as :

$$MSE = r_{dd}(0) + \mathbf{w}^T\mathbf{R}\mathbf{w} - 2\mathbf{p}^T\mathbf{w} \qquad (2.15)$$

For the searching of the optimum $\mathbf{w}^O$, instead of calculating the exact least square, a gradient descent method is used. The gradient vector which is a column vector is obtained by:

$$\nabla = \frac{\partial(MSE)}{\partial \mathbf{w}} = \left[\frac{\partial(MSE)}{\partial w_0} \quad \frac{\partial(MSE)}{\partial w_1} \quad \cdots \quad \frac{\partial(MSE)}{\partial w_{L-1}}\right]^T$$

$$= 2\mathbf{R}\mathbf{w} - 2\mathbf{p} \qquad (2.16)$$

Since the global minimum MSE is obtained where $\nabla = 0$, the Eq. 2.16 can be rewritten as:

$$\mathbf{w}^O = \mathbf{R}^{-1}\mathbf{p} \qquad (2.17)$$

To update the weight for each iteration of adaptation, the Eq. 2.16. is multiplied by $\mathbf{R}^{-1}$ and yields following equation:

$$\tfrac{1}{2}\mathbf{R}^{-1}\nabla = \mathbf{w} - \mathbf{R}^{-1}\mathbf{p}$$

$$\mathbf{w}^O = \mathbf{w} - \tfrac{1}{2}\mathbf{R}^{-1}\nabla \qquad (2.18)$$

According to the Eq. 2.18, we should calculate $\mathbf{R}^{-1}$ and ∇. The problem for this task is that the matrix $\mathbf{R}$ is unknown, and ∇ must be estimated using local statistical information. It means that the actual value cannot be obtained, and only the estimation value can be found at each iteration of adaptation.

Since ∇ and $\mathbf{R}^{-1}$ are estimated, the modified version is introduced. For obtaining more stable behaviour, a factor μ is used to smooth the prediction value of $\mathbf{R}^{-1}$ and gradient ∇ :

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu \mathbf{R}^{-1} \nabla_k \tag{2.19}$$

2.2.1 Least Mean Square (LMS)

The crosscorelation matrix $\mathbf{R}$ may not be invertible, and if even is theoretically invertible, the numerical precision required to invert $\mathbf{R}$ is beyond the capability of the computational system (Messerschmitt et al., 1986). The LMS algorithm is derived to tackle this problem and produces the simpler adaptive algorithm without calculating the inverse of $\mathbf{R}$. It is obtained by simplifying the estimation of the $\mathbf{R}^{-1}$:

$$\mathbf{R}_k^{-1} = \delta \mathbf{I}_N \tag{2.20}$$

where δ is a positive constant called the adaptation step size, and the $\mathbf{I}_N$ is the identity matrix of order N. Eq. 2.19 can be written in a simpler form as :

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu \hat{\nabla}_k \tag{2.21}$$

When the MSE is minimum, the weight is optimum and it is denoted by $\mathbf{w}^O$. Let J be defined as the cost function of the system or the quality criterion of the system. In this case MSE is used as the quality criterion.

$$\Delta J = J - J_{\min} \tag{2.22}$$

Therefore for all position in the MSE surface:

$$\Delta J > 0 \quad \text{for } \mathbf{v} \neq 0, \text{ where } \mathbf{v} = \mathbf{w}^O - \mathbf{w}$$

The goal is to find the $\mathbf{w}^o$, the weight vector for which the performance criterion J is minimised. Suppose $\mathbf{w}$ starts with an initial value, and the algorithm will choose the next value which is closer to the $\mathbf{w}^o$. When updating is performed and ΔJ is still greater than 0, the next improvement should be continued. The iteration will stop until ΔJ reach ε, where ε is the convergence error.

One method to search for $\mathbf{w}^o$ is by employing the gradient descent technique. By evaluating the derivative of the performance function J, the weight adaptation can move from the current value to the next value in order to improve the performance function J at time index k. Since MSE is used as the performance function J, the derivative of J can be written as :

$$\nabla_k = \frac{\partial(J)}{\partial \mathbf{w}_k} \tag{2.23}$$

To simplify, the prediction error can be used to estimate the gradient of J:

$$\hat{\nabla}_k = \frac{\partial[e^2(k)]}{\partial \mathbf{w}_k} = 2e_k \frac{\partial e_k}{\partial \mathbf{w}_k}$$

$$e_k = d_k - \mathbf{w}_k^T \mathbf{x}_k$$

$$\hat{\nabla}_k = 2e(k)\frac{\partial}{\partial \mathbf{w}_k}\{d(k) - \mathbf{w}^T\mathbf{x}(k)\} = -2e(k)\frac{\partial}{\partial \mathbf{w}_k}\{\mathbf{w}^T\mathbf{x}(k)\}$$

$$= -2e(k)\mathbf{x}(k) \tag{2.24}$$

The weight adaptation for LMS can be written as:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e(k)\mathbf{x}(k) \tag{2.25}$$

This algorithm has been popular for a long time since it was introduced by Widrow (Widrow et al., 1960). The step by step of LMS algorithm can be formalised as:

Step 1. The prediction value of output is calculated.

$$y(k) = \mathbf{w}^T(k)\,\mathbf{x}(k) \tag{2.26}$$

Step 2. The error with respect to the desired signal $d(k)$ is calculated.

$$e(k) = d(k) - y(k) \tag{2.27}$$

Step 3. The coefficients of the filter is updated according to:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\, e(k)\, \mathbf{x}(k) \tag{2.28}$$

This algorithm has been widely used, and there are many applications using this approach such as echo removal (Kuo and Zhao, 1990), adaptive interference cancelling (Treichler et al., 1986). However, this algorithm has some limitations. Firstly, the initial convergence is slow. Secondly, the convergence of the system depends on the input signal characteristics, due to the dependence of $e(k)$ on the input signal. Finally, a residual error still exists after convergence because the weight adjustment will oscillate around the optimal value.

There are some modifications from the standard LMS algorithm. Those modifications are the Griffith Algorithm (Griffith, 1967) and the Sign Error LMS Algorithm (Gersho, 1984).

2.2.2 Recursive Least Square (RLS)

The RLS algorithm tries to approach the last adapted weights as the overall optimal weights. It is different from the LMS algorithm,. which will oscillate about the converge point, rather than actually converging to the optimal point. In the LMS algorithm, $\mathbf{R}^{-1}$ in Eq. 2.19 is assumed to be $\mathbf{I}$, but the RLS algorithm make uses of the estimate of $\mathbf{R}$ in Eq. 2.19:

$$w_{k+1} = w_k - \mu \hat{\mathbf{R}}_k^{-1} \hat{\nabla}_k \tag{2.29}$$

By estimating **R**, the algorithm starts with initial **R** and initial w. The original RLS algorithm is:

Step 1. The autocorrelation matrix **R** is updated via :

$$\mathbf{R}_{k+1} = \mathbf{R}_k + \mathbf{x}(k)\,\mathbf{x}^T(k) \tag{2.30}$$

Step 2. The crosscorelation vector **p** is updated via :

$$\mathbf{p}_{k+1} = \mathbf{p}_k + d(k)\,\mathbf{x}(k) \tag{2.31}$$

Step 3. **R** is inverted to produce the matrix $\mathbf{R}^{-1}$

$$\mathbf{R} \rightarrow \mathbf{R}^{-1} \tag{2.32}$$

Step 4. The weight w can be updated by using **P** and $\mathbf{R}^{-1}$

$$\mathbf{w}^{O}_{k+1} = \mathbf{R}^{-1}_{k+1}\,\mathbf{p}_{k+1} \tag{2.33}$$

The algorithm requires the inversion matrix of the matrix **R**, that leads a high computational complexity, i.e. it needs N^3+2N^2+N multiplications. To simplify the updating process, some refining methods have been developed. For example using (ABCD) Lemma (Kailath, 1980) the inverse of **R** can be calculated successively:

$$\mathbf{R}^{-1}_{k+1} = \mathbf{R}^{-1}_{k} - \frac{\mathbf{R}^{-1}_{k}\mathbf{x}(k)\mathbf{x}(k)\mathbf{R}^{-1}_{k}}{1+\mathbf{x}^T(k)\mathbf{R}^{-1}_{k}\mathbf{x}(k)} \tag{2.34}$$

From this equation, $\mathbf{R}^{-1}_{k+1}$ is not calculated directly by inverting the **R**

The implementation of the RLS algorithm is obtained by making use of the adaptation gain vector z(k), which is defined by expressing the gain as a function of the input signal only, and which can be updated using a set of adaptive prediction filters. The weight adaptation for RLS algorithm can be expressed in as:

$$\mathbf{w}^{O}_{k+1} = \mathbf{w}^{O}_{k} + \frac{e(k)\cdot \mathbf{z}_k}{1+q} \tag{2.35}$$

where $e(k)$ is the a priori error :

$$e(k) = d(k) - y_0(k)$$

$y(k)$ is the a priori output :

$$y_0(k) \triangleq \mathbf{x}^T(k)\mathbf{w}_k^0$$

z is the filtered information vector or the adaptation gain vector :

$$\mathbf{z}_k \triangleq \mathbf{R}_k^{-1}\mathbf{x}(k) \tag{2.36}$$

q is the normalised input power :

$$q = \mathbf{x}^T(k)\mathbf{z}_k \tag{2.37}$$

Eq. 2.35 is used by the RLS algorithm for updating the weights at each iteration. It start with $\mathbf{w}_k^0$ and then updates the weights according to the input signal $x(k)$ and the desired signal $d(k)$. The algorithm updates the w value in order to find the optimal value of w. The adaptation of w is controlled by three terms, $e(k)$, $\mathbf{z}_k$, and q.

The first term of adaptation, is the a priori error $e(k)$, which is calculated by subtracting the desired signal $d(k)$ with the a priori output $y(k)$. The a priori output is the output of the adaptive filter which is calculated using the previous optimal weights or the weights that have not been updated at this iteration. This a priori error is known as the prediction error. If the prediction error is equal to zero, updating will not be performed.

The second factor is $\mathbf{z}_k$, the adaptive gain vector. By multiplying **R** to vector x(k), the direction and length of vector x(k) is influenced by the matrix **R**. Therefore is z is the modification version of the input vector x(k) by **R**.

The last term is q which represents the input signal power. The input signal power $\mathbf{x}^T(k).\mathbf{x}(k)$ is normalised by $\mathbf{R}_k^{-1}$. This normalisation yields the input signal power average, rather than being proportional to the actual signal power.

However these three terms produces exactly the optimal weight update. And since $\mathbf{R}_k$ is not negative, 1+q always equal or more than 1. A more efficient RLS algorithm can be formalised as follow:

Step 1. A new input sample $x(k)$ is accepted and shifted into the $\mathbf{x}(k)$,

$$x(k) \rightarrow \mathbf{x}(k) \quad ;d(k)$$

Step 2. The a priori output is computed.

$$y_0(k) = \mathbf{w}_k^{ot}\mathbf{x}(k)$$

Step 3. The priori error is computed

$$e_0(k) = d(k) - y_0(k)$$

Step 4. Filter information vector is computed

$$\mathbf{z}_k = \mathbf{R}_k^{-1}\mathbf{x}(k)$$

Step 5. The normalised error power is computed

$$q = \mathbf{x}^t(k)\mathbf{z}_k$$

Step 6. The gain constant is calculated

$$v = \frac{1}{1+q} \tag{2.38}$$

Step 7. The normalised filtered information vector is computed

$$\mathbf{z}_k = v \cdot \mathbf{z}_k \tag{2.39}$$

Step 8. The optimal weight vector is update

$$\mathbf{w}_{k+1}^o = \mathbf{w}_k^o + e_0(k)\tilde{\mathbf{z}}_k \tag{2.40}$$

Step 9. The inverse correlation matrix is updated by using the Eq. 2.41.

$$\mathbf{R}_{k+1}^{-1} = \mathbf{R}_k^{-1} - \tilde{\mathbf{z}}_k\mathbf{z}_k^T \tag{2.41}$$

For each pair of the input-desired signal $\{x(k), d(k)\}$, the computational complexity of the RLS algorithm to calculate the output and the update of weight is $2N^2+4N$ multiplications, $2N^2+4N$ additions, and 1 division. Since this amount is used for one pair of input and desired signal, the total of multiplication over N-1 to L-1 is

$$C_{RLS} = (L-N+1)\cdot 2N^2 + (L-N+1)\cdot 4N \qquad (2.42)$$

The complexity is lower than the direct method, and does not require a matrix inversion calculation. The RLS algorithm has been successfully implemented for many applications such as dynamic modelling. The problem with this algorithm is the initialisation matrix, and for some problems, the calculation of **R** becomes numerical unstable.

2.3 Application of Adaptive Filters to Signal Enhancement Problems

Speech transmission and processing are often degraded by acoustic or electrical noises. The objective of speech enhancement is to improve the speech quality by pre-processing or post-processing. Noise reduction is a time series problem which requires a dynamic setting. In the DARPA Neural Network Study, Section 19.6.1 Recovery of Noise-Corrupted or Distorted Waveforms, the following statements are made (Hoyt et al., 1990):

> "A classical signal processing problem is that of recovering an analog signal after transmission over a noisy or dispersive channel. In many cases there may [be] very little knowledge about the standard characteristics of the signal, noise or dispersion. Standard

approaches to this problem include filtering for noise reduction and channel equalization to reduce dispersion, and the application of estimation theory to form an optimal estimate to the equalisation to reduce the dispersion, and the application of estimation theory to form an optimal estimate of the desired waveform. The lack of knowledge or relevant statistical characteristics hinders the use of estimation theory."

The speech enhancement process is required to improve the quality and intelligibility of the received signal. Some problems which require a speech enhancement are echo (Gee and Rupp, 1991), background noise (Hansen and Clements, 1991), transmission noise (Soleit et al., 1988), inter-symbol interference (Biglieri et al., 1984), fading (Coloma et al., 1991), howling (Kuo and Chen, 1992). The unavailability of a clean signal as an ideal signal model or the channel model makes this problem hard to solve by using a conventional filter which is designed by using a predefined filter response (Classen et al., 1983).

In speech enhancement, conventional signal processing has focused on compensating for distortions introduced by channel variations. Various methods have been proposed to tackle this problem including fixed filtering (Boll, 1979) and adaptive filtering techniques using conventional linear adaptive filters. Non-linear techniques have been developed as well, but these typically depend on a local linearisation of the problem. The use of adaptive methods has improved the performance of background noise elimination to a large extent (Vary, 1983). In these methods, the parameters of the filter are changed to adapt to the non-deterministic characteristics of the background noise. However, provision has to be made for the

filter to be able to change in the environment. Sometimes, it may lead to unsatisfactory performance of the overall system (Connel et al., 1990).

Most of the conventional linear adaptive filters employ the LMS or RLS algorithms. However, the LMS and RLS algorithms have several limitations. Firstly they are designed to deal with the linear problem, since the output function is a linear combiner. Therefore, even though the filter performs adaptation, the output always follows Eq. 2.1. It means that the LMS or RLS performs piecewise linearisation to deal with non-linear problem. Secondly, for producing the best result, the filter needs thousand of tap-delays. It increases the amount of calculation and the amount of delay. The adaptive filter only produce an output after thousand of input samples. Thirdly, since the LMS and RLS algorithms are derived by using assumptions that the signal is stationary, the filter cannot attack the non-stationary problem effectively (Shynk, 1992).

The failure of Adaptive Noise cancelling to adequately track the microphone spacing transfer function leads the use of a Neural Network as an arbitrary reference primary mapping element. (Connel, 1990). There have been a number of Multilayer Perceptron Architecture to deal with this problem. Waibel et al. (Waibel et al., 1989) used a time delay neural network architecture. In this architecture there is no feedback from a later layer to a previous layer. Another approach involves feedback from a hidden layer to the output layer (this is known as a recurrent network). A third approach is to use an intermediate structure between a global feedforward and a global recurrent architecture to yield an overall global feedforward with a local recurrent nature. This time delay network model uses Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) synapses derived from a linear adaptive filter.

Back and Tsoi (Back and Tsoi, 1991) state that the FIR models will give better results than the IIR models. By combining a resonator-banks filter and an associative memory neural network, an adaptive filter has been developed by Sztipanovitz (Sztipanovitz, 1990).

Artificial Neural Networks (ANNs) offer an alternative technique for adaptive filtering. Gorman and Sejnowski (Gorman and Sejnowski, 1988) show that an ANN requires far less restrictive assumptions about the structure of the input signal than any traditional techniques. A general ANN model for adaptive filtering has been proposed by Nerrand et al. (1993). It has also been shown that an ANN can be applied to separate a signal into different signals (Cohen et al., 1991) and to perform signal classification (Malkoff, 1992). In the area of background noise elimination, Xue et al. (Xue et al., 1992) has applied an ANN-based adaptive matched filter for biomedical processing applications.

There are similarities between the ANN model and the conventional adaptive filter model as mentioned by Marcos et al. (Marcos et al., 1992). An adaptive filter uses the gradient descent technique for adaptation of the filter coefficients, and this is also the common way of adjusting the weights of the connections in a supervised neural network. The significant difference between the neural network approach and the conventional adaptive filter is that the neural network normally synthesises a non-linear function of its inputs, as opposed to the linear function of the adaptive filters. ANNs are inherently non-linear models, and ANN-based filtering methods are potentially useful for signals with inherent non-linearity.

Parameter estimation is an important part of an adaptive filter system. A traditional self-tuning adaptive control system with Recursive Least Square is

sensitive to noise and can only deal with a linear system (Chow et al., 1990). By using a neural network, the parameter estimation process can be performed to reject the noise in the non-linear system. It shows that a neural network has a great potential in system identification as mentioned by Chow et al. (Chow et al., 1990). A neural network which is formed as a three-layer backpropagation model can be trained to be able to identify the pole and zero of the system. According to Kanekar et al. (Kanekar et al., 1990), by applying a Madaline net, and by using independent random noise as input, the network can perform parameter estimation better than using a Kalman filter method. It is shown that by applying the associative characteristics of neural networks, they can handle signal estimation faster and more accurately than the conventional system.

The signal detection problem can be modelled as a pattern classifier problem, and some neural network models can be used as pattern classifiers. A Kohonen feature map is one such model. Moreover, Adaptive Resonance Theory via an unsupervised neural network can deal with plasticity-stability dilemma, but will perform poorly when the input pattern is coloured by noise (Lippmann, 1987).

Conventional adaptive filtering techniques have proved valuable in signal processing problems where training data are available. However, for non-linear problems, the solution is difficult. Therefore, it leads to the development of adaptive filter using Artificial Neural Network. By employing the ANN techniques, less assumptions are required to build the model, because the ANN has capability to learn the input-output relation in the non-linear relationship. It is indicated that in the problems where sufficient data are available for training, it may be useful to develop a trainable system. The use of ANN introduces non-linear processing that adds more

flexibility to the system capability, which may lead to better performance than conventional linear adaptive systems in certain applications.

In Chapter 3, the basic model of an Artificial Neural Network combined with a Fuzzy System (FS) are described. The formal model of the ANN, which is a non-linear processing model, is explained followed by the discussion of FS as the model-free estimator which can deal with vagueness. Both techniques are used as the basic framework for designing a novel adaptive filter.

Chapter 3

Artificial Neural Network and Fuzzy System

An Artificial Neural Network (ANN) model is an alternative form of information processing that is fundamentally new and different information-processing paradigm. It is fast becoming an established discipline. However, it does not replace algorithmic programming, because on a philosophical level, they are not compatible. They complement each other nicely. This chapter describes the theoretical framework of the Artificial Neural Network model, followed by the description of the Fuzzy System (FS). Both systems are used as the basic structure to build the proposed model.

3.1 Artificial Neural Network

ANNs are good at some tasks, e.g. solving complex pattern recognition, understanding continuous speech, identifying hand-written character, for what conventional algorithms are poor. ANN models have been considered particularly suitable for unstructured computations, such as pattern recognition, artificial intelligence problem solving, and approximation to large optimisation problems (Abu-Mostafa, 1989). The difference between ANNs and the other approaches in cognitive science and artificial intelligence is the inference process and the knowledge base in the ANN model is non-separable. In ANNs, the algorithms blur the distinction between data and program, and the algorithm is represented in their

architecture while their data are represented implicitly by the dynamics of ANN model (Mehra and Wah, 1992).

In formal terminology, ANN are concerned with non-programmed adaptive information processing systems, that develop associations (transformation or mapping) between objects in response to the environment. Instead of being given a step-by-step procedure for carrying out the desired transformation, an ANN generates its own internal rules governing the association, and refines those rules by comparing its results to some examples (Hecht-Nielsen, 1988a).

An ANN processes immense quantities of information in parallel and are inspired by models of brain behaviour. Hecht-Nielsen stated the following definition of artificial neural system (Hecht-Nielsen, 1988b):

> "A neural network (NN) is a parallel, distributed information processing structure consisting of processing elements (which can possess a local memory and carry out localised information processing operations) interconnected together with unidirectional signal channels called connections. Each processing element has a single output connection which branches ("fan out") into as many collateral connections as desired (each carrying the same signal-the processing element output signal). The processing element output signal can be of any mathematical type desired. All of the processing that goes on within each processing element must be completely local; i.e., it must depend only upon the current values of the input signal arriving at the processing element via impinging connections and upon values stored in the local memory".

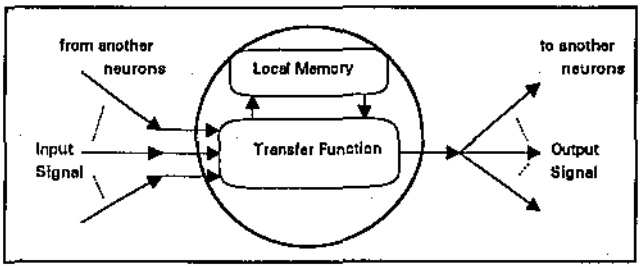

Figure 3.1 A neuron structure (Hecht-Nielsen, 1988b)

Structurally a neural network can be defined as a non-linear directed graph with weighted edges that are able to store patterns by changing the edge weights and are able to recall patterns from incomplete and unknown inputs (Simpson, 1987). The transfer function of a neuron structure as shown in Figure 3.1 is a non-linear transformation. Non-linearity of the neural network increases the richness and facilitates noise suppression (Kosko, 1990). However, it also produces computational risk and analytical intractability. Furthermore in some conditions, the system can enter into dynamic instability.

An ANN is a dynamic system that consists of a large number of simple processing elements connected in parallel (Vassiliadis, 1990). They may be proved to be more robust when distributions are generated by a non-linear process and are strong non-Gaussian (Lippman, 1987). An ANN model is specified by the functional description of the connections of network, network topology, neuron characteristics, and learning mechanism. These will be discussed in the following section.

3.1.1 Formal description of a neuron

Basically a artificial neural network system consists of a number of interconnected neurons. To formalise the description of an artificial neural network following

definition are used. This definition follows the general model definition as stated by Lee (Lee, 1991):

DEFINITION 3.1

A discrete time neuron can be specified by 5 tuple :

$$N = \{F, q, \mathrm{w}, \Gamma, C\} \tag{3.1}$$

The components of a neuron and the interactions between each components are shown in Figure 3.2.

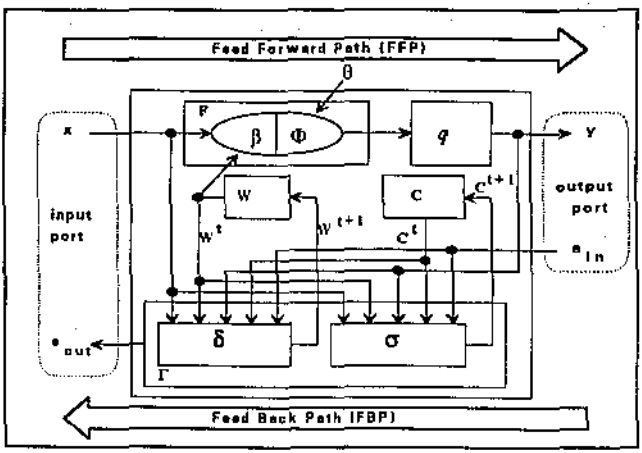

Figure 3.2 A formal neuron

Each neuron (as a generic processing unit) functions on a discrete time scale, and it operates on n-tuples of input $\mathbf{x} = [x_1, \ldots, x_n] \in \mathrm{R}$ and produces a real scalar y as the output of a neuron. This value is propagated through the network via unidirectional connection to other neurons in the network. The set of input arguments $\mathbf{x} = \{x_1, x_2, \ldots x_n\}$ is called the **Receptive Field** (RF) of this neuron.

Signal flow path in a neuron

The signal flow path from the input port through to the output port is called the **Feed Forward Path** (FFP) and the signal flow path from the output port through δ to the input port is called the **Feed Back Path** (FBP).

At any time index t, for the neuron N, the value presented in the RF of neuron N is designated by $IN(N)^t$. The output of the neuron N designated by $OUT(N)^t$, is equal to the state value $q(N)^t$ of neuron N. Therefore, for the neuron N, $IN(N)^t = x^t$ and $OUT(N)^t = q(N)^t$.

e_{in} is called the **Received Feedback Signal** (RFBS) and is the feedback signal received from the output port of Neuron N. The signal e_{out} is called the **Transmitting Feedback Vector** (TFBV) and is the list of feedback signals, which are called the **Transmitting Feedback Signals** (TFBS). These feedback signals are transmitted upstream to the RF of neuron N. There is one to one correspondence between the variables in RF and the components of e_{out} and the dimension of IN is equal to the dimension of e_{out}.

Feed Forward Path Component

For the feed forward path, the components that are used are : State Transition Function,F and Composite State, q. Those components are described in following discussion.

State Transition Function (STF)

F, is called the State Transition Function (STF) and is a parameterised function with w as the parameter vector of the function. F is used to generate the Composite

State, q of this neuron with $q = F(\mathbf{x}|\mathbf{w})$, where the vector $\mathbf{x} = (x_1, x_2, ..., x_n)$ designates the list of input arguments for F. The function F itself is a triple:

$$F = \{\beta, \Phi, \theta\} \tag{3.2}$$

This first mapping is mathematically described by a basis function β. Further transformation is performed by a non-linear activation function Φ, together with the external threshold θ, to yield a new activation value y. The final output y can usually be expressed as :

$$y = \Phi(\beta(\mathbf{x}|\mathbf{w}), \theta) \tag{3.3}$$

Basis Function (β)

$\beta : R^n \rightarrow R$ is a basis function, which is a delayless function and is normally in a very simple form. It may be a linear basis function or a radial basis function. A linear basis function (LBF) which is a hyperplane-type function that is a first order linear basis function. The output value of this function is a linear combiner of the inputs.

$$\beta(\mathbf{x}|\mathbf{w}) = \sum_{j=1}^{n} w_{ij} x_j \tag{3.4}$$

A radial basis function (RBF) is a hypersphere-type function and is a second order (non linear) basis function. The output value of the basis function represents the distance of the input vector to the reference vector.

$$\beta(\mathbf{x}|\mathbf{w}) = \sqrt{\sum_{j=1}^{n} (x_j - w_{ij})^2} \tag{3.5}$$

Weight (w)

$\mathbf{w} = [w_1, ..., w_n] \in R^n$ is a weight vector of a neuron. This n-tuples of weight values are associated with each connection. **w** is called the **Modulating Parameter**

Structure (MPS) for F, because each parameter w in $\mathbf{w}$ is used to modulate the function F. The collection of all possible values of $\mathbf{w}$ for a neuron N forms a vector space, called the **Neuron Parameter Space** for the ANN. The property of this space is important for analysing the parameter adaptation (or learning process) of a neuron.

Neuron decision rule (Φ)

$\Phi : \mathrm{R} \rightarrow \mathrm{R}$ is the neuron decision rule. The function Φ itself is a non linear function and it may be a step, ramp, sigmoid or Gaussian function. The most widely used is sigmoid for LBF network and gaussian function for RBF network. For LBF model, the type of neural decision function used only influences the speed of learning not the accuracy of the network (Kalman et al, 1992).

Sigmoid function is defined in following equation and shown in Figure 3.3:

$$\Phi(u) = \frac{1}{1 + e^{-\frac{u}{\sigma}}} \tag{3.6}$$

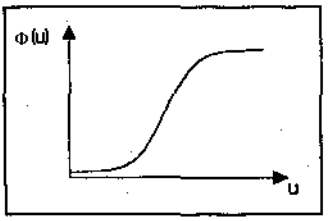

Figure 3.3 Sigmoid function

Gaussian function is stated in Eq 3.7 and depicted in Figure 3.4.:

$$\Phi(u) = ce^{-\frac{u^2}{\sigma^2}} \tag{3.7}$$

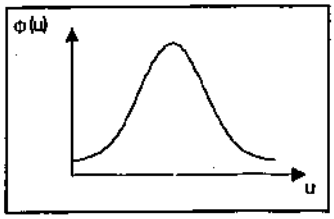

Figure 3.4 Gaussian function

Real Threshold (θ)

$\theta \in R$ is a real threshold which may be supplied from external or by fixing it with a constant value.

Therefore, the final output or the activation value of the neuron y is

LBF neuron : $$y = \Phi\left(\sum_{j=1}^{n} w_j u_j - \theta\right) \tag{3.8}$$

RBF neuron : $$y = \Phi\left(\sqrt{\sum_{j=2}^{n} (w_{ij} - x_j)^2} - \theta\right) \tag{3.9}$$

Composite State (q)

q is the **Composite State** that represents the states of a neuron. The static state of a neuron is given by its last output value y. The dynamic state of the neuron is described by the first-order difference equation that governs the time evolution of the y value or the neuron output. The time is not included as an independent variable and it is assumed to be fast at the neuron level, i.e. during time index t to time index $t+1$ there is not any change of the neuron's state. The dynamic state is only determined by q^{t+1} and q^t.

Feed Back Path Component

For the feedback path, the component which are used are: Parameter Adaptation Automata (PAA) and Parameter Control State (PCS). The feedback path is occurred after there is a feed forward path that produce the Composite State (q) this value is used by the feed back path components.

Parameter Adaptation Automata (Γ)

Γ is called the **Parameter Adaptation Automata** (PAA) of a neuron and it is specified by a pair

$$\Gamma = \{\delta, \sigma\} \tag{3.10}$$

Where δ is the Parameter Adjusting Function and σ is the Parameter State Transition Function.

Parameter Adjusting Function (δ)

δ is called the **Parameter Adjusting Function** . It is known as learning function of neuron N and is a mapping that takes $\mathbf{x}^t$, $\mathbf{w}^t$, q^t, $\mathbf{C}^t$, $\mathbf{e}_{in}^t$ as inputs to generate $\mathbf{w}^{t+1}$, i.e.,

$$\mathbf{w}^{t+1} = \delta(\vec{x}^t, q^t, \mathbf{w}^t, e_{in}^t, C^t) \tag{3.11}$$

Parameter State Transition Function (σ)

σ is called the **Parameter State Transition Function** (PSTF) of neuron N and is characterised by the following equation:

$$C^{t+1} = \sigma(\vec{x}^t, q^t, \mathbf{w}^t, e_{in}^t, C^t) \tag{3.12}$$

Parameter Control State (C)

C is called **Parameter Control State** (PCS) of neuron, is a structure of variables used to control the parameters adjusting process (learning process) of this neuron.

A neuron which does not contain e_{in} and e_{out} in *C*, is called a **Uni-Directional Neuron** (UDN). Formal neurons with both e_{in}, and e_{out} specified are called **Bi-Directional Neurons** (BDN). Normally, BDN is used in supervised learning. The neurons in the backpropagation network are examples of this category.

The neuron output *y*, in neural modelling is known as the **Short-Term Memory** (STM) of the network. Normally, a neuron will forget the previous value as soon as there is a new input signal. This storage mechanism is very short time. The weights of the network **w**, store the knowledge of the system. They encode the **Long Term Memory** (LTM) of the pattern information (Kosko, 1990).

A neuron operates in two phases of operation: the learning phase and the recall phase. During the learning phase, a neuron adjusts the **w** values according to the training set and the learning rules in order to optimise a particular quality criterion. The adjustment of **w** values is performed by the function δ (Parameter Adjusting Function), and controlled by *C* (Parameter Controlled State). *C* itself changes during the learning phase, that is done by σ (Parameter State Transition Function). Not all types of neuron lea.ning process require *C* to be specified. A learning mechanism with PCS specified is called a learning mechanism with internal state, while a learning mechanism without PCS specified is called a memoryless learning mechanism.

When a neuron is performing the recall phase, Γ and *C* are not used any more, or they are set to fixed values. In addition, for most of ANN architectures, Γ

and C are the same for the entire neurons in the network, respectively. Therefore, Γ and C can be pulled out from the single neuron model and be abstracted outside a neuron model. Thus, it controls the entire neurons in the ANN model. It produces the simpler neuron model but less general than the neuron model stated above. According to the Eq. 3.2, the key element of a neuron in recalling operation is an accumulation of information by a non-linear decision rule which operates on comparison between the accumulated information and a threshold. Therefore, in simplifying the model in the recalling process the C, q, and Γ can be omitted from the single neuron model to produce the model depicted in Figure 3.5.

The inhibitory input is the input that tries to inhibit the neuron, i.e. has the negative connection, and the excitatory input is the input that activates the neuron, i.e. it contributes to the value that exceed the threshold value.

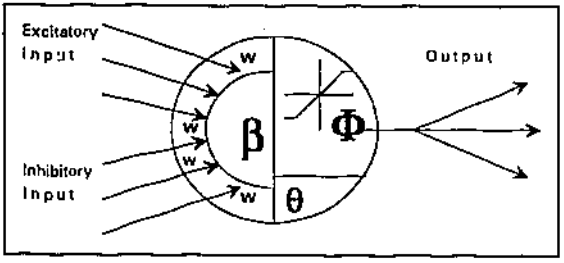

Figure 3.5 Process in neuron.

This definition of a formal neuron is general and it covers most of the current neural network models. In performing more complex tasks, a single neuron is not sufficient and a number of neurons have to be arranged in a such way to build more complex system. This arrangement can be done in the same level or/and in different

level. Each neuron in an ANN model is connected to another neuron with th synaptic weight w.

DEFINITION 3.2

A neural network consists of a collection of neurons interconnected in such a way that the output of each neuron functions as input to any sub-collection of neurons. Each neuron in the network receives one input from the output of the other neurons in the network, respectively. These connections may be excitatory or inhibitory connection. A set of neural input and output are identified as the networks input and networks outputs, respectively. The network state is the collection of individual neuron states

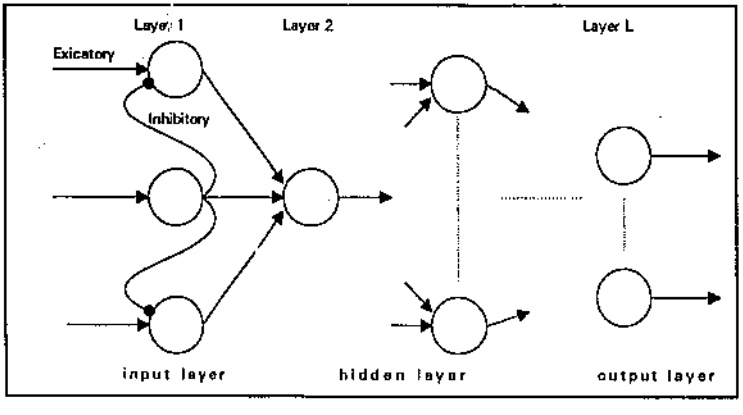

Figure 3.6 Neuron connection

According to Definition 3.2, the architecture of an artificial neural network can be formalised and specified by the set:

$$G = \{A_1, A_2, ..., A_n\} \tag{3.13}$$

Where A_1 , A_2, ... A_n are **Network Elements** (NE) and each network element is defined by a triple:

$$A_i = [N_i, R_i, T_i] \tag{3.14}$$

where N_i is the **Processing Kernel (PK)**, R_i is called **Input Connection Map** (ICM) and T_i is called the **Output Connection Map**. This formalisation of the r ' network architecture that is uses Eq. 3.13 and Eq. 3.14 are used to defin proposed model in the Chapter 4.

N_i of A_i and can be either a neuron or a neural network. R_i for A_i contains the fan-in interconnection information. The set of variables connected to N_i through R_i is called the **Receptive Field** (RF) of A_i. T_i for A_i contains the fan-out interconnection information from A_i. Furthermore, the set which contains all the variables connected through T_i to the output of a NE A_i is called the **Projective Field** (PF) of A_i.

The arrangement of the neurons can be performed in different levels known as multi-layer neuron structure. This structure provides more complex function (Cotter, 1990) or more complex decision region (Sethi, 1990) of the ANN model in the overall system. Lippman (Lippman, 1987) has shown that a three-layer network is adequate to form a complex decision regions which is required to solve the non-separable problem in the real application, i.e. for the non-separable classification task (Murphy, 1990)

DEFINITION 3.3

An L-layer feedforward neural network is built by L ordered sub-collections of neurons called layers with interconnections specified as follows: for l = 2, ... , L the inputs of the neuron at the l-th layer are obtained from the outputs of the neurons at

the (l-1)-th layer. The inputs to the first layer (called the input layer), are the **network inputs**. The set of neurons with some input arguments received from the inputs of the network is called the **Input Set** (I-Set) of the network or is known as the **Input Layer** of the neural network model. Each neuron in the I-Set is called a **Neural Input Element** (NIE) of the network. The outputs of the L-th layer are the **network outputs**. The set of neurons with output sent to the output port of the network is called the Output Set (O-Set) of the network and this set is know as **Output Layer** of the neural network model. Each neuron in the O-Set is called a **Network Output Element** (NOE) of the network. Layer 2 to L-1 are called **hidden layers**.

A connection between each neuron at adjacent layers may be feedforward connection, feedback connection, lateral connections, or time delayed connection. In feedforward connection, data from a lower layer are propagated forward to neurons of an upper layer via feedforward connections. The feedback connections bring the output of a neuron of an upper layer back to neurons in the lower layer. A lateral connection is the connection of neuron at the same layer (Leemon, 1991). Time-Delayed connection uses a delay element in the connection to yield temporal dynamic of that connection (Werbos, 1989).

The signal in the ANN model can flow in both directions, i.e. from input layer to output layer or from the output layer to the input layer. For the feedforward path, the inputs to the network are the signals extracted from the external variables which are connected to the input ports of the network input elements or the input layer of the network. The outputs of the network are the signals generated by the network output elements or the output layer and are connected to the variables external to the

network. During the feed-forward operation, R defined in Eq. 3.14 assigns to each input argument of N, one of the output from other A's in G, or one of the input signals to the network. During the feedback operation, T defined in Eq. 3.14 assigns to each Received Feed back Signal (RFBS) of N, one of the Transmitting Feedback Signal from some A's in G or one of the RFBS of the network.

3.1.2 Feature mapping in neural network

Most of the current ANN models are multilayer neural network architectures. In feedforward multilayer networks, there is the natural concept of a sequential flow of information from each layer to the subsequent upper layer. From each layer to the subsequent layer, a feature mapping process is performed. It means that each layer expresses a different internal representation of a specific feature of the input data (Abu-Mostafa, 1989). At the input layer, a neuron value represents the data that are still raw data, and in moving to higher layer, the neuron value expresses the higher representation of the input data. This gradual transformation from raw data to higher representation is very interesting, especially if the representation evolves spontanteneously via the learning mechanism. However, for some ANN architectures, the exact representation at each layer is still being investigated. Basically, this feature mapping between each layer is the transformation function of each neuron. The mapping networks can be classified into two classes (Hecht-Nielsen, 1987). They are feature based networks and prototype based network.

Feature based networks

These networks implement a functional input/output relationship that is expressed in terms if general, modifiable functional form $H_n = \{ h_1 (.), h_2(.), \ldots , h_n(.)\}$ which is a set of the fixed family functions. For a given neural network architecture (with n inputs), each specification of weight and threshold for the neuron by applying one or more learning rules within the processing element of network, gives rise to one function in H_n. The modification of weight is done to find the specific mapping that is to be approximated, by determining $h : X_n \rightarrow Y_n$ and $h \in H_n$. Examples of this kind of networks are : Backpropagation (Rumelhart et al., 1986), Functional Link Neural Net (Pao, 1989), GRNN (Poggio and Girosi, 1990), Cascade Correlation (Fahlman and Lebiere, 1990).

Prototype based networks

These networks create a set of prototypes $((v_1, w_1), (v_2, w_2), \ldots , (v_L, w_L))$ of input-output examples $((x_1, y_1), (x_2, y_2), \ldots , (x_k, y_k))$ that statistically represent the function being approximated. This process is performed by employing one or more learning rules. The network produces the mapping on a new unknown vector by comparing the input vector with the set of prototype vectors stored in the networks. By using the result of a similarity comparison, for example minimum distance or maximum activation value, the ANN produces output vector as an estimate of $f(x)$. Examples of this kind of networks are counterpropagation network (Hecht-Nielsen, 1987) and MAXNET network (Lippman, 1987). Most of these ANNs in this class use the binary hidden units, as the winner nodes or the chosen prototypes.

For a three-layer ANN model which uses binary hidden units, the existence of J-1 hidden units is the necessary and sufficient condition for given J input patterns. It

has been proved by Arai (Arai, 1989) that a three-layer network with infinite hidden units and the activation function is absolutely integrable has the outputs are arbitrary for the continuous value. It has also been shown that the existence if the infinite hidden units is not only the sufficient condition, but also the necessary condition if the activation function for the hidden units are discrete at most of countable points.

3.1.3 Computation and learning in neural network

According to Venkatesh (Venkatesh, 1992), any problem of finite dimensions is computable by a neural network. In the implementation of an ANN model for solving a problem, the ANN system performs two main computational mechanisms which are the learning mechanism, and the recall mechanism or operation mechanism.

The neurons and the neuron connections in an ANN model are considered as the hardware of the system, and the weights and threshold can be considered as the software. In order to program the ANN system, a set of weights which simulate the computation that will be performed have to be chosen. If this process of choosing the weight and threshold values can be automated, it will constitute a learning mechanism.

Let **G** be an architecture of an ANN model which is defined in Eq. 3.13. Then learning is any change in any component of **G** :

$$\dot{\mathbf{G}} \neq 0 \qquad (3.15)$$

Those changes may be performed to each component of **G**, the N_i parameters of each neuron, the connection strengths and the connection directions R_i and T_i, and the size of matrix for self growing neural network, i. It means that the changing can affect

the architecture of the ANN and the value of each weight and parameter of a neuron. Therefore, learning cannot be performed without changing (Kosko, 1991).

The concept of learning by examples is depicted in Figure 3.7. The learning mechanism builds an ANN that implements an arbitrary function f, when given a sufficient a number of input-output examples from that function. This mechanism is performed automatically as dictated by the learning algorithm. The learning process commences with a set of training examples in pairs of input and expected output i.e. $\{(\mathbf{x}_1, f(\mathbf{x}_1)), (\mathbf{x}_2, f(\mathbf{x}_2)), \ldots, (\mathbf{x}_k, f(\mathbf{x}_k))\}$. During the learning phase, the changing of neural networks configuration is made by adjustment of the weights and thresholds, so as to make the network simulate a function f. It is performed with the goal to make the response of the network to the input $\mathbf{x}$ as close as possible to the desired output $f(\mathbf{x})$. The implementation may be only a good approximation of the function f. This mechanism will eliminate the need to redesign a new network architecture each time there is a new function to be implemented by the ANN.

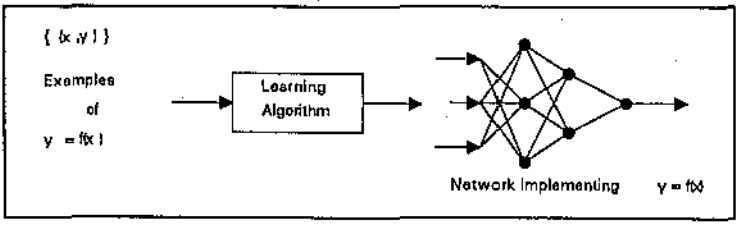

Figure 3.7 Learning mechanism concepts

Different learning algorithms require different types of information from the learning environment and employ different heuristics rules for the weight and threshold modifications. Each learning mechanism operates in an information environment. The processing elements are fed by incoming signals. An ANN is a distributed

information processing structure and each processing element is totally independent. All the processing elements can respond according to its transfer function applied to the incoming signals and performs self-adjustment in response to their local information environment.

The interaction of an ANN with its environment is used to classify the learning paradigms. The environment presents the input patterns to the ANN and may generate a feedback in reaction to the network's output. If feedback is prescriptive, that is if a desired output is specified for every training input, then the learning mechanism is classified as supervised learning. Supervised learning incorporates an external teacher and/or global information. If the feedback is evaluative, that is if only an evaluation of the output is provided, then the learning is classified as reinforcement learning. Unsupervised learning does not require any feedback. The learning can still adapt in order to satisfy some internal objectives. Unsupervised learning involves no external teacher and relies upon only a local information and internal control. It self-organises according to the presented data and discovers the emergent properties of data. It means that an unsupervised learning scheme will try to perform a property recognition of the input data without a definition (Simpson, 1992).

The learning process in an ANN model based on a set of examples can be regarded as synthesising a multidimensional function which belongs to a problem of hypersurface consisting sparse data points. It simply means that the learning is the collecting process of the input coordinates $\mathbf{x}$, and the corresponding output values $f(\mathbf{x})$ at those input coordinates. It builds an adaptive look-up table of input-output relation. This problem is equivalent to the associative memory that retrieves the

appropriate output when presented with the input and uses generalisation to produce an output when presented a new input. It is also similar to the problem of estimating a system that transforms inputs into outputs given a set of examples of input-output pairs. A generalisation can be performed by interpolation, which estimates a location $f(d)$ in space $(x, f(x))$ after d is fed into the system where there are no examples i.e. no data about the location d and $f(d)$. Interpolation is the limit of approximation. The capability of an ANN to perform interpolation shows the capability of the ANN in the plasticity problem. The plasticity problem arises whenever an ANN is supplied with unknown input, or the data which have not been used for the training data (Carpenter and Grossberg, 1988).

A multivariate function $f(\mathbf{x})$ is approximated or interpolated by an approximating function $H(\mathbf{x}|\mathbf{w})$ having a number of parameters $\mathbf{w}$ and $\mathbf{x}$ as the input vector, where $\mathbf{x}$ and $\mathbf{w}$ are real vector. For a specific $H(\mathbf{x}|\mathbf{w})$, the learning process tries to find the set of parameters $\mathbf{w}$ that provide the best possible approximation of f on a set of examples $\{(\mathbf{x}_1, f(\mathbf{x}_1)),(\mathbf{x}_2, f(\mathbf{x}_2)), \ldots ,(\mathbf{x}_k, f(\mathbf{x}_k))\}$. Therefore, it is very important to choose the type of an approximation function H that can represent f as close as possible. There would be little point in trying to learn if the chosen approximating function $H(\mathbf{x}|\mathbf{w})$ could only give a very poor representation of $f(\mathbf{x})$ even with optimal parameter values.

In measuring the quality of approximation, a distance function ρ is usually used. It determines the distance between the approximation $H(\mathbf{x}|\mathbf{w})$ to $f(\mathbf{x})$. Let $f(\mathbf{x})$ be a continuous function defined on a set $\mathbf{x}$, and $H(\mathbf{x}|\mathbf{w})$ an approximating function defined continuously on $\mathbf{w}$. The approximation problem is to determine the optimal parameters $\mathbf{w}^o$ such that:

$$\rho(H(\mathbf{x}|\mathbf{w}^{O}),f(\mathbf{x})) \leq \rho(H(\mathbf{x}|\mathbf{w}),f(\mathbf{x})) \tag{3.16}$$

for all **w**. The solution of this problem is said to be the best approximation. The number of examples needed to approximate a function reasonably well grows exponentially with the ratio between the dimensionality and its degree of smoothness ρ.

In performing the construction of the hypersurface which approximates f by using $H(\mathbf{x}|\mathbf{w})$, the current $\mathbf{w}^t$ value will be updated to the next value $\mathbf{w}^{t+1}$. Generally the updating rules for the parameters **w** is :

for unsupervised learning :

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \varepsilon^t L(\mathbf{x}^t, \mathbf{w}^t) \tag{3.17}$$

for supervised learning :

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \varepsilon^t L(\mathbf{x}^t, \mathbf{y}^t, \mathbf{w}^t) \tag{3.18}$$

where L is the function of the learning rule and it is equivalent to Γ in Eq 3.8 and $\mathbf{w}^t$ denotes the weights of ANN in time t and consists of a set of **w** for each neuron. Therefore, **w** represents the components T_i and R_i in Eq. 3.14 for each neuron in the ANN. The vector **x** is the input vector, and **y** is the desired vector, and the ε is the modification step. The function L is the gradient of the cost function or a heuristic rule. In the learning process, a learning scheme moves from one state to the next state as to optimise the weights according to a particular cost function C, where $L = \nabla C$. This movement has the goal to reach the global minima of that cost function (Battoud and Gallinari, 1992).

Let x be an instance of a concept to learn. This concept is defined by a probability density function $p(x)$ and w represent the parameters of the learning system. For a given state of the system, $J(x,w)$ is a local cost function, which

measures how well the system behaves on x. The goal of learning is to optimise the global cost function C. The expectation of local cost function over the space $\mathbf{X}$ of the concept of learning usually is written as :

$$C(w) = \int_X J(x,w)p(x)dx = E_x\{J(x,w)\} \tag{3.19}$$

Most often the explicit form of $p(x)$, and $C(w)$ are unknown. The information which is available comes from a series of examples of the variable x. Therefore, the realisation of $J(x,w)$ is possible by a measurement based on the observation of $\{x_i\}_i$. A necessary condition of optimality for the parameters of the systems is :

$$\nabla C(w) = E_x\{\nabla_w J(x,w)\} = 0 \tag{3.20}$$

where ∇ is the gradient operator. Since $C(w)$ is unknown and only the realisation of $J(x,w)$ is available, the classical optimisation methods cannot be used. One solution is to apply adaptive algorithm:

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \gamma^t \nabla_w J(\mathbf{x}^t, \mathbf{w}^t) \tag{3.21}$$

where γ is the gradient step. This modification strategy is similar to the technique that has been described in the Chapter 2.

In learning by example there are two main problems : an information problem and a complexity problem.

The information problem is in the terms of generalisation. Under what combinations will the performance of the network on the set of examples persist on previously unseen input? The learning algorithm will construct f from only partial information on the input and output relation of f, i.e. a number of examples $y = f(x)$. It is clear that there are cases when the examples are not enough to cover all the information about f. In this situation, the algorithm cannot be expected to produce

an implementation of something it does not know. On the other hand, even if the examples contain enough information about *f*, the complexity this information may be prohibitive. For example, suppose there are only a few examples but a large network is used. In this case, it cannot be expected to reach any good generalisation. In general, for a fixed set of examples, a smaller network will perform better generalisation although it does not mean that a small network is more likely to implement the function better. It is only more likely to behave similarly on an unknown input which never be represented in the set of example. The other conflicting requirement is that the network should be large enough to handle the function being implemented, regardless the generalisation question.

The complexity problems can be posed in the terms of polynomial time complexity. Under what conditions does there exist an algorithm that runs reasonably fast in time that is polynomial in the size of problem to produce a network implementation of *f* from the set of examples. Most of learning tasks run sufficiently fast for small problems. However, when the problem size increases the computation time scales poorly. It is consistent with the theoretical prediction. The complexity of learning has been studied and most of the results indicate that the learning complexity may be prohibitive (Abu-Mostafa, 1989). The representation of data is crucial to the complexity of problem. In feedforward network, there are several internal representations of the data at each layer. This transformation from the raw data to higher level representation is very interesting in the research on ANNs, especially if the representations are derived spontantenously via the learning mechanism.

Thus the learning scheme in a mapping neural network is a process of constructing the hypersurface function from sparse data point by optimising the cost function in order to reach the global minima. The result of the training yields a network which simulates the implemented function as close as possible according to the quality criterion which is used for the cost function.

3.2 Fuzzy System

Uncertainty of the information such as the dynamic or the static state of a complex system has been a problem in the designing an adaptive system. To deal with uncertainty many paradigms such as probability or possibility can be used (Dubois, 1993). In order to measure the uncertainty of information or vagueness, the entropy concept is used (Klirr and Folger, 1988). Fuzzy approach which builds up from the possibility theory arises from the ambiguity or the vagueness (Black, 1973) and has been applied in many areas such as control system (Lee, 1990a), pattern recognition (Lim et al., 1992), speech processing (Jiangxin, 1992), social science (Taber, 1994), medicine (Klirr and Folger, 1988), management and decision making (Kaufmann and Gupta, 1988), and virtual reality (Kosko and Dickerson, 1994).

3.2.1 Fuzzy set

The concept of fuzzy set was introduced by Zadeh (Zadeh, 1965) in his seminal paper. It illustrated the simplicity and the significance of fuzzy sets in dealing with the vagueness. Zadeh's definition of a fuzzy set is stated in the following :

> Let X be a space of points (objects), with a generic element of X denoted by x. [X is often referred to as the universe of discourse]. A

fuzzy set (class) A in X is characterised by a membership (characteristic) function $m_A(x)$ which associates with each point in X a real number in the interval [0,1], with the value of $m_A(x)$ representing the "grade of membership" of x in A. Thus the nearer the value of $m_A(x)$ to unity, the higher the grade of membership of x in A.

DEFINITION 3.4

If X is collection of objects, then a fuzzy set A is a subset of the universe of discourse X and the membership is defined as the ordered pair :

$$A = [x, m_A(x)], \tag{3.22}$$

where $x \in X$ and $0 \le m_A(x) \le 1$. The membership function $m_A(x)$ represents the degree to which the object x belongs to the set A where $m_A(x) = 0$ indicates that x does not belong to the set A, and $m_A(x) = 1$ represents full membership. It is the degree to which the deterministic measurement x is compatible with the vague concept of A.

$$m_A(x) = \text{Degree}(x \in A) \tag{3.23}$$

In Figure 3.8, suppose the objects are x_1 and x_2, and the membership function to set A is defined by $m(x)$. The degree of object x_1 belongs to set A is m_1 and the degree of object x_2 belongs to set A is m_2.

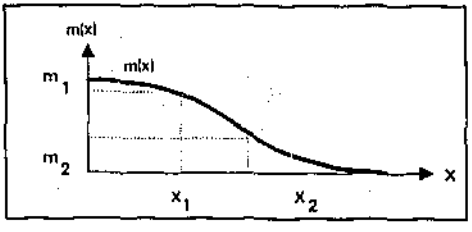

Figure 3.8 Membership function

Kosko has extended the concept of the elementhood into a subsethood (Kosko, 1989). The subsethood explains the degree of a set belonging to another set. It provides more capability for the fuzzy set approach to deal with vagueness.

Fuzzy set operations are similar to the Cantor set operations such as AND, OR, and NOT (Zadeh, 1965). It has been improved by Kosko by adding another operations such as subsethood, supersethood and fuzzy entropy (Kosko, 1990).

3.2.2 Fuzzy system components

A Fuzzy System (FS) allows imprecise representation of the system rather than forcing the use of precise statement to describe the system transformation. There are four principal elements in a fuzzy system: fuzzification interface, fuzzy rule base, fuzzy inference machine, and defuzzification interface (Wang, 1992). The interconnection of them are shown in Figure 3.9.

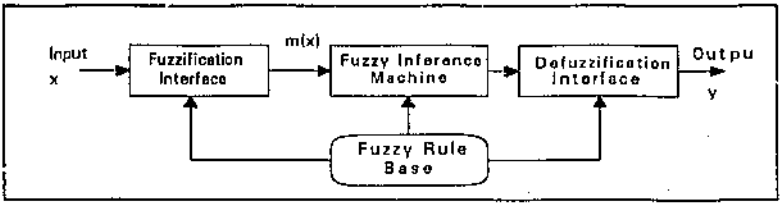

Figure 3.9 Building block of fuzzy system

The fuzzification interface

This element describes the vagueness of the input signal by mapping from the observed input universe of discourse X to the fuzzy set defined in X. Let A be a fuzzy set defined in X and x an input to the fuzzification interface. Then the outputs of the fuzzification interface is $m_A(x)$.

There are two factors that should be considered in the fuzzification interface. They are the number of fuzzy sets and the specific membership functions used.

- The number of fuzzy sets defined in the input universe of discourse.

Let $\mu_i = 1, 2, ..., n$, be the number of fuzzy sets defined in the i' th subspace of X, where the i'th subspace of X is the projection of X onto the i'th coordinate of R^n, i. e., it is the set $\{x_i : x = (x_1, x_2, ..., x_n) \in X\} \subset R$. The number of fuzzy sets determines the complexity of the fuzzy system. This complexity includes time complexity, i. e. the computational requirement, and the space complexity, i.e. the storage requirement. Both complexities generally increase with the number of fuzzy sets, because the system produces more number of membership values of each fuzzy set when an input is fed into the fuzzification interface. On the other hand, more membership values increase the accuracy of inference. Therefore, there is always a trade-off between accuracy and the complexity of the system.

- The specific membership functions for these fuzzy sets.

Let $\mu_{A_i^j}, i = 1, 2, ..., n, j = 1, 2, ..., m_i$ are the membership functions of the j'th fuzzy set defined in the i'th subspace of U. Those membership functions can be expressed in one of the following formats:

Gaussian membership function:

$$\mu_{A_i^j}(x_i) = a_i^j \exp\left(-\frac{1}{2}\left(\frac{x_i - \bar{x}_i^j}{\sigma_i^j}\right)^2\right) \qquad (3.24)$$

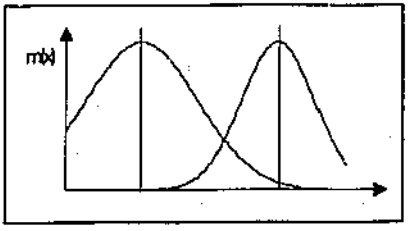

Figure 3.10 Two Gaussian membership function

Triangle membership function:

$$\mu_{A_i^j}(x_i) = a_i^j \left(1 - \frac{\left| x_i - \bar{x}_i^j \right|}{\sigma_i^j} \right) \tag{3.25}$$

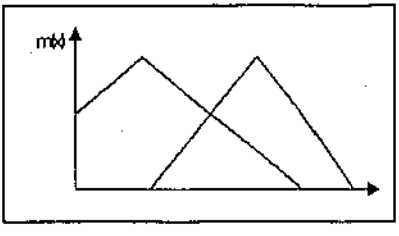

Figure 3.11 Two triangle membership function

The triangle membership function is the most widely used (Asakawa and Takagi, 1994). Since the value a_i^j can be set to be equal to 1 for every membership function, it has only two main parameters $\bar{x}_i^j$ and σ_i^j :

$$\mu_{A_i^j}(x_i) = f\left(x_i \left| \bar{x}_i^j, \sigma_i^j \right. \right) \tag{3.26}$$

This membership function influences the smoothness of the input-output relation. In general, the sharper membership function is, the less smooth is the input-output surface. The choice of the membership function shape is still being investigated, and currently it is still determined by the application (Cox, 1992).

The fuzzy rule base

A rule is made up of two parts: the antecedent or the premise, and the consequent. The antecedent can contain many *clauses* linked by the logical operator AND, OR, and NOT. Clauses can be further decomposed into *variables* and *adjectives*. These elements build a set of linguistic statements in the form of:

"IF a set of conditions are satisfied THEN a set of consequences are inferred"

In fuzzy set approach, each adjective of a clause is assigned a membership function. Given a value for input variable, the membership function is used to calculate the truth value of that variable and each clause in antecedent can be assigned a fuzzy truth value. The next step is to assign a truth value to the entire antecedent. Fuzzy truth value is the degree by which a rule is relevant to a measured situation. It describes the degree by which the rule's antecedent is true in a fuzzy sense. The fuzzy truth value of a rule depends upon the fuzzy truth value of each of its clauses and upon the logical operators linking them. The consequent maps the truth value of the antecedent to the output space.

The rules in the fuzzy rule base provide a natural form in which humans represent their knowledge. There are many forms of fuzzy rules (Lee, 1990b), in this work, only the following form in Eq. 3.27 is shown below:

$$R_j: \; IF \;\; x_1 is A_1^j \;\; and \;\; x_2 is A_2^j \;\; and \ldots and \;\; x_n is A_n^j \;\; THEN \;\; z is B^j \qquad (3.27)$$

where x_i=(i = 1,2, ... , n) are the inputs to the fuzzy systems, and z is the output of the fuzzy systems. A_i^j and B^j are the linguistic terms, and j is the number of fuzzy rules in the fuzzy rule base. The premise space is a n-dimensional and the input $x_1, x_2, \ldots, x_n$

is partitioned into the fuzzy subspaces. This partitioning is called premise structure in a fuzzy model. As a result the identification process in designing a fuzzy system implies determining how the input space should be partitioned.

The fuzzy rules may be gathered from two sources: human experts, or a set of training data. A general method to generate fuzzy rules from numerical data has been proposed by (Wang and Mendel, 1992b). In designing the number of fuzzy rules in a fuzzy rule base the specific statement of each fuzzy rule has to be considered as a design parameter.

The fuzzy inference machine

This element performs the decision making logic which employs fuzzy rules from the fuzzy base to produce the fuzzy output as a function of the fuzzified inputs to the fuzzy system. It is similar to a human decision making procedure based on fuzzy concepts and linguistic statements. There are many different kinds of fuzzy logic operations which may be used in a fuzzy inference machine. Therefore, the decision on which specific fuzzy logic is used is an important factor in designing a FS. In this work, the inference process is not performed separately but performed together with the fuzzification input by the fuzzy neuron.

The defuzzification interface.

The final step is to combine the output values of each of the rules in a manner that reflects the truth value of each rule. This element transforms the fuzzy output of a fuzzy system to produce a non-fuzzy output. There are 30 defuzzification techniques (FAQ, 1994). This process actually quantises the fuzzy output into a non-fuzzy output. In addition to the defuzzication technique which is applied, in designing a FS

system, the number of fuzzy sets defined in the output universe discourse X and specific membership functions of these fuzzy sets have to be considered. The most important requirement is that the defuzzication has to reflect the truth value of each rule.

Among the defuzzification techniques, the centroid which is based on the centre of mass method seems to provide the best performance for most applications (Cox, 1992). The centroid defuzzification technique can be written as:

$$\bar{y} = \frac{\int_{-\infty}^{\infty} y\,\mu(y)\,dy}{\int_{-\infty}^{\infty} \mu(y)dy} \tag{3.28}$$

Where y is the output value for each rule, μ is the fit value of each rule to the input and the $\bar{y}$ is the final output of defuzzification interface.

In the discrete case, it can be rewritten as :

$$f(x) = \frac{\sum_{j=1}^{K} (\bar{z}^j \mu^j)}{\sum_{j=1}^{K} (\mu^j)} \tag{3.29}$$

Where $\bar{z}^j$ is the output of j^{th} rule, μ^j is the fit value of the j^{th} rule, x is the input and $f(x)$ is the final output of the fuzzy system. It is shown that a fuzzy system is a complicated non linear system which maps a non-fuzzy value in $U \subset R^n$ into the non-fuzzy value in R. A fuzzy system works with parallel associative inference. When an input is given, a fuzzy system fires each rule in parallel, but to a different degree to infer a conclusion or output. Thus, a fuzzy system reasons with sets, fuzzy or multivalued sets, instead of bivalent propositions.

In summary, a fuzzy system has the some design parameters : number of fuzzy sets defined in the input and output universe of discourse, membership functions of these fuzzy sets, number of fuzzy rules in the fuzzy rule base, linguistic

statements of the fuzzy rules, decision making logic used in the fuzzy inference machine, and defuzzification method. Therefore, in building a FS system by using a set of training data, the building process has to be done in two phases. The first phase is the structure identification phase, which must solve two problems: finding input variables and finding the input-output relationship. The second phase is the parameter identification, that is the determination of the membership functions of the fuzzy set (Sugeno, 1992).

3.3 Model-Free Function Approximator.

An ANN or an FS basically performs a mapping from input space to the output space, and it means that the system performs a mathematical function. As stated by the existence theorem from Kolmogorov (Cotter, 1990), an ANN model can implement a continuous function (Hecht-Nielsen, 1989) by using a three-layer ANN model. As proved by Wang and Mendel (Wang and Mendel, 1992a), a fuzzy system (FS) can be used as a universal approximator as well, and separately Kosko (Kosko, 1992a) has described the capability of additive fuzzy systems to perform approximation of any continuous function on a compact domain to any degree of accuracy.

ANNs and FSs are model-free estimators (Cox, 1992). Unlike the statistical estimator, they estimate a function without knowing the mathematical model of the input-output relation. They learn from experience which is represented as numerical or sometimes linguistic data samples. Both ANN and FS show the intelligent capability by adaptively estimating continuous function from data without specifying mathematically how outputs depends on the inputs (Kosko, 1993). Mathematically,

both ANN and FS transform input signal to output signal, and this transformation defines the system itself and determines the characteristics of the system.

3.3.1 Function approximation by neural network

A function f, denoted $f: X \rightarrow Y$, maps an input domain X to an output range Y. For every element x, in the input domain X, the function f uniquely assigns the element y in the output range Y. This assignment is denoted as $y = f(x)$, which is a function that defines the causal hypothesis. A mapping network simulates a function $F : X \rightarrow R$ defined on $X \subset R$ by expressing the function being simulated by using a finite combination of additions and superposition with real functions of one variable. Any family $F_W : X \rightarrow R^m$ of mapping network parameterised by $\mathbf{w} \in \mathbf{W}$ will be called a mapping network architecture. A given architecture takes input from an input space X_n (n represents the dimension of the input space) and produces the corresponding outputs in an output space Y_m (m represents the dimension of the output space).

In an ANN, the mapping function is built up by training by examples, which is a collection of data $\{(x_i, y_i), 1 \leq i \leq k\}$ i.e. $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$, where $y_i = f(x_i)$. Therefore, it is assumed that there is a set of examples which are available for training, and the aim is to produce a function g that approximates function f, as close as possible, so that $g(x_i) \approx y_i$. For any given $\varepsilon > 0$, the function f is said to be approximated within ε by function g and denoted it by $f \equiv g$, if and only if $|f - g| \leq \varepsilon$.

Specifically, the mapping neural network approximates a bounded mapping or function $f: A \subset R^n \rightarrow R^m$, from a compact subset A of an n-dimensional Euclidean space to a bounded subset of $f[A]$ in an m-dimensional Euclidean space by means of training by examples $(x_1, y_1), (x_2, y_2) \ldots (x_k, y_k)$ where $y_i = f(x_i)$.

This is quite similar to classical interpolation and estimation problems, where a function has to be found from a class of functions that interpolate the data as close as possible. One of the promising features of neural networks is that the mathematical model of the mapping is learned and the empirical data obtained from the application serves to define the problem without a formal mathematical model being specified. The concepts of approximating arbitrary function using functional forms that do not depend upon either orthogonatility or linear superposition turn out to be an important theme in neurocomputing (Hecht-Nielsen, 1988a), because any reasonable function can be computed by some neural networks (Venkatesh,1992).

The ability to perform a function approximation is ensured by the Kolmogorov theorem, as stated by Kolmogorov's Mapping Neural Network Existence Theorem (Hecht-Nielsen, 1987). This theorem states that for any integers n and m, given any $\varepsilon > 0$ and any function $f : [0,1]^n \subset R^n \rightarrow R^m$, where $f(x) = y$, f can be implemented exactly by a three-layer feedforward network having n fan-out processing elements in the first layer, $(2n + 1)$ processing elements in the middle layer, and m processing elements in the output layer. Thus, f is approximated within ε mean square error accuracy in the family of three-layer feedforward neural networks comprised of sigmoid neurons in the first two layers and linear neurons in the output layer. Even though the theorem is proved for a continuous vector mapping on the unit cube $[0,1]^n$, it can be extended to apply to any compact when required, i.e. closed and bounded. Cybenko (Cybenko, 1990) has shown that by using continuous (but not necessarily monotonic) sigmoidal function the function approximation can be achieved. Specifically, given an arbitrary collection of input-output data, the above theorem implies that by using a network with sufficiently many hidden nodes,

all at one layer, the network parameters can be chosen to achieve any desired modelling error criterion. Therefore, a three-layer perceptron with infinite number (continuum) of computing elements is able to represent any absolutely integrable function on R^n (Kowalczyk, 1991). This theorem shows that it is possible to perform a continuous function approximation by a neural network. However, this theorem is strictly an existence theorem, it does not provide a constructive explanation of the method for developing the network itself. Although this result shows the universality of ANNs with only a single hidden layer, the determination of the number of nodes is still a major question.

Most of the models for function approximation are feedforward models. The backpropagation neural network model has been able to perform function approximation better than traditional method in the accuracy of time series prediction problem (Jones et al. 1989). In this problem, a certain number of points of a time series are given to the system and the values of the time series at some future time will be predicted. The backpropagation network has been used in the prediction task of the sunspot series, department store sales data and stock index time series (Srirengan and Looi, 1991).

However, there are some problems in employing a backpropagation neural network for the function approximation problem. Firstly, it requires a great deal of training data that covers most of the domain. Secondly, the interpolation is poor without a great deal of training i.e. the number of iterations has to be enough to make the system converge in the adaptation. Thirdly, it is much slower for comparable accuracy that the best non-neural network methods (Farmer and Sidorowich, 1987). In some cases, large amount of data are not available therefore a function

approximator should be able to interpolate and extrapolate from a small data set. Furthermore, in many applications the learning process must occur in real time and the slow learning in the backpropagation networks restricts the application areas of neural nets.

To increase the speed of learning, several neural network models have been developed. By using a Radial Basis Function Network (RBF-Net) to replace the sigmoidal non-linear elements (Moddy and Darken, 1989), a significant improvement in speed has been achieved. However, it only addresses the problem of excessive training data or poor interpolation. To achieve comparable accuracy more neurons and more number of training data points are required. Some modification of the RBF-Net has been developed to improve the interpolation and to reduce the amount of training necessary. This modification is obtained by applying a normalisation and known as Connectionist Normalised Linear Spline Net (CNLS) (Lee, 1989, Jones 1989, Howel et al., 1989).

The advantage over traditional methods of function approximation is that the network can be used as the development tool and the complex mathematical interactions between network nodes can provide a better model of process than more traditional approaches. This approach has been implemented in an embedded system for wind direction and wind speed determination system (Farley, and Varhol 1992). In this application, the conventional method such as Kalman Filter cannot be used because the Kalman Filter method assumes white noise and an auto regressive moving average (ARMA) time series model which cannot be taken for a this application.

3.3.2 Function approximation by fuzzy system

The function approximation problem has been attempted by using fuzzy system by Wang and Mendel (Wang and Mendel, 1992b). However, the rule assignment is performed manually.

The combination of fuzzy basis function method with the conventional method such as Gram-Schmidt Orthogonal Least squares to determine the significant basis functions and the remaining patterns has been developed (Wang and Mendel, 1992c). It provides a one-pass regression procedure. Therefore, it is much faster than the backpropagation algorithm. Also, it yields a more robust system which is insensitive to noise in its input. The combination of linguistic and numerical input provides important advantages over polynomials and radial basis function. Due to the fact that a linguistic input from the human expert can be encoded into the system. It has been proved that a fuzzy basis function is capable of approximating any real function (Wang and Mendel, 1992a). This model has been applied to solve non-linear problems.

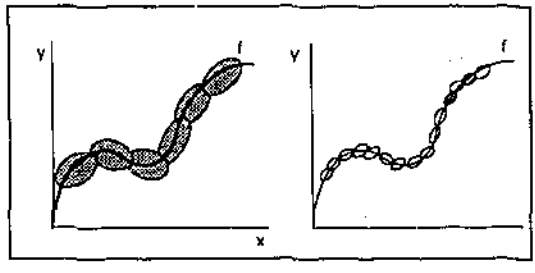

Figure 3.12 Function approximation by fuzzy patch (Kosko, 1992)

In Fig, 3.12. it is shown that a function $y = f(x)$ creates a line. Each value on x-axis is mapped to one or more values on y-axis by the fuzzy patch which is represented by

the shaded area. A fuzzy system creates overlapping fuzzy patches along this line. The fuzzy patches in the input-output product space $X \times Y$ cover the function $f: X \rightarrow Y$. The smaller and the crisper the fuzzy patches, the more they resemble the line and produce better approximation of the function f. When they become infinitely small and numerous they converge to the line.

A fuzzy rule defines a fuzzy patch in the input-output space. In the conventional fuzzy systems the fuzzy rules are derived from experts. More recently, by using a statistical approach or a neural network, the fuzzy rules can be derived from the sample data (Munakata and Jani, 1994). Different experts or algorithms generate different sets of fuzzy rules. Some techniques to derive the fuzzy rules systematically has been investigated, most of them using optimisation technique such as RLS (Wang and Mendel, 1992c), membership adaptation (Dasarathy, 1992), Genetic Algorithm (Machado et al., 1992), or Neural Network (Mitra et al., 1994).

If the function is simple i.e. $y = x$, there is usually no reason to apply the fuzzy patch approach, it may do perfectly well using conventional approach and fuzzy logic could worsen the result (Mc Neill and Freiberger, 1993). However, when the function cannot be modelled mathematically, especially when the function is non-linear, discontinuous, or non-differentiable, the function can be divided and solved with fuzzy patches. Fuzzy patches can also be easier to derive and faster to use. Moreover, when a control situation cannot be formalised, an engineer simply does not know the exact function. Yet machine operators may know its approximate course by the vague rules of thumb they use to guide the system. Therefore, fuzzy IF-THEN patches can deal with this situation (Kosko, 1993).

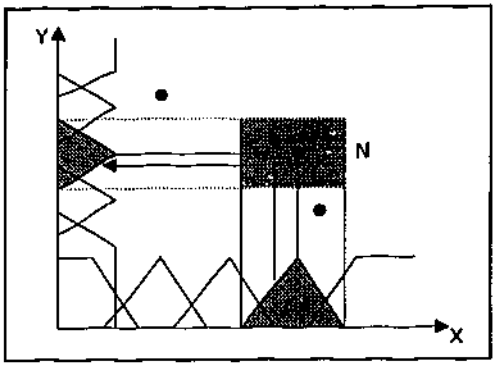

Figure 3.13 Fuzzy patch as fuzzy rule (Kosko, 1993).

In approaching a function, an ANN model use numerical point samples (x, y) to build the function that approximates the actual function. In contrast, a FS estimates the function by using the fuzzy set samples. Both kind of samples reside in the same input-output product space X × Y (Kosko, 1993). In Figure. 3.13, the numerical point sample (x, y) is represented as dot. The fuzzy set sample as the membership function at each axis, and the fuzzy set association is a rule and it is represented as a box *N*. The fuzzy set sample (X,Y) encodes the structure of the fuzzy rule. It represent a mapping of minimal fuzzy association from a part of the input space to a part of the output space. The x-axis represents the antecedent part as the input associant and the y-axis represents the consequence part of the fuzzy rule as the output associant Y. In Figure 3.13, the box is the fuzzy patch which behaves as a fuzzy rule. The association also represents the result of an adaptive clustering algorithm.

In general, a fuzzy system S maps families of fuzzy sets to families of fuzzy sets, thus:

$$S: I^{n_1} \times \ldots \times I^{n_r} \rightarrow I^{p_1} \times \ldots \times I^{p_s} \qquad (3.30)$$

Where I^n is the fuzzy set in each input variable and I^p is the fuzzy set of the output variable. This work focuses on fuzzy system that maps hyper-sphere of fuzzy sets in I^n to hyper-sphere of fuzzy set in I^p:

$$S: I^n \rightarrow I^p \tag{3.31}$$

Thus, this FS behaves as an associative memory. The FS maps the closest input to the closest outputs. One successful implementation of this FS system is Additive Fuzzy Associative Memory (Kosko, 1989).

As proved by Wang (Wang, 1992), Jang (Jang, 1992) and Kosko (Kosko, 1992) by using the Stone-Weierstrass theorem, there exists a FS that can be used for approximating an arbitrary non-linear continuous mapping to any accuracy. This existence theorem is similar to Kolmogorov theorem in the ANN model.

Since a neural network with learning capability can reduce the development time and cost of the designing a fuzzy system (Asakawa and Takagi, 1994), a combination of both techniques may result a learnable system that can deal the vagueness. Adaptive Network based Fuzzy Inference System (ANFIS) applies neurons which employs AND and OR operations. This fuzzy neural model is a highly non-linear mapping model. Therefore, it is superior to linear methods in reproducing non-linear time series (Jang et al., 1993). However, this architecture is still employing backpropagation type gradient descent learning (Jang, 1992). Furthermore, the ANFIS model requires initial parameter settings which have to be able to capture the underlying dynamics of the function. The RBF-Net, that is based on the hypersurface fitting technique, has gained increasing popularity in many practical areas such as pattern recognition, signal processing, system modelling, and control system. It is due to its simple structure, well-established theoretical basis, and fast learning. Nie

and Linkens (Nie and Linkens, 1993) have shown that there are some similarities between RBF-Net and Fuzzy control algorithm in approximating a function. The fuzzy approach which is applied by using RBF-Net has been applied for controlling the blood pressure in hospital intensive care unit (Nie and Linkens, 1993).

In the Chapter 4, a combination model of Fuzzy System and Artificial Neural Network is described. This model is designed to be applied to adaptive filter tasks by exploiting the function approximation capability of the Fuzzy System and the learning capability of the Artificial Neural Network.

Chapter 4

Fuzzy-CPN Model

This chapter addresses the development of a fuzzy counterpropagation network (Fuzzy-CPN) which has fast learning capability in order to perform a continuous function approximation in an adaptive filter. As mentioned by Rumelhart et al. (Rumelhart et al., 1986) in their seminal paper on the neural network model, there are eight major aspects :

- A set of processing units , i.e. the set of A_i in the set G in Eq. 3.13.
- A state of activation, the interpretation of y in Eq. 3.3.
- An output function for each unit F in Eq 3.1 including the function Φ and β in the Eq. 3.2.
- A pattern of connectivity among units, i.e. the set of components R_i and T_i of each A_i in Eq. 3.14.
- A propagation rule for propagating patterns of activities through the network of connectivities, i.e. the direction of flows x, y, e_{in} and e_{out} in a neuron, and the flows of the information in T_i and R_i in the Eq. 3.14 in the overall of G.
- An activation rule for combining the inputs impinging on units with the current state of that unit to produce a new level of activation of the unit.
- A learning rule whereby patterns of connectivity including the δ and σ in the Eq. 3.10, are modified by experience.
- An environment within which the system must operate.

In order to introduce this proposed model, all of those aspects of an artificial neural network are described in this chapter. In addition, as the system is used as an adaptive filter, the ANN architecture that is designed must satisfy some application considerations of the system :

- Learning must be performed in a minimal number of iterations of training.
- Since the computation resource is limited, the network must be able to adapt the structure of neural network to optimise the memory.
- The processing time should be considered to achieve a fast processing in the neural network and filtering, i.e. the structure of architecture can be implemented in a parallel way.

4.1 Proposed Model

Artificial Neural Networks (ANN) and Fuzzy System (FS) are similar in many ways and share a more formal property and the same state space (Wassermann, 1993). They are continuous vector mappers from the input space to the output space and can generalise in producing a correct response despite minor variations in the input vectors. ANN and FS are model-free estimators which learn from experience and encode their information in a numerical parallel-distributed framework.

The ANN and FS process inexact information and process it inexactly. A number of uncertainty problems arise in the neural network's input data, propagation of result through the network, and interpretation of final result (Cohen and Hudson, 1992). The ANN model recognises ill-defined patterns without explicit specifications set of rules. The ANN acquires knowledge through training but it cannot take the advantage of an expert's presence, and statistical neural estimators require a

statistically representative sample set. In contrast, an FS estimates functions and controls the system with partial description of system behaviour, and can use the knowledge from an expert to solve the problem and derive the linguistic rule of the fuzzy system.

However, there are fundamental differences between these techniques. An ANN model and a FS differ in how they estimate the sample function. These differences appear during the system construction. The neural approach requires the specification of a non-linear dynamic system, the acquisition of a sufficient representative set of numerical training samples, and the encoding of those training samples in the dynamic system by repeating the training cycles. A FS requires only partial information of the system in a linguistic rule matrix (Kosko, 1993). This task is simpler and faster to be solved than designing and training a neural network. After the system is constructed, the same numerical input to both system can be applied. The output resides in the same numerical space of alternatives and both systems define a surface or manifold in the input-output product space $X \times Y$.

The combination of both approaches has been developed by many researchers such as the fuzzy neural network with fuzzy signals and weights (Hayashi et. al., 1993) which is used for fuzzy expert system, fuzzy hierarchical analysis, fuzzy modelling to derive the rules. Fuzzy Basis Functions (FBF) can be used for universal approximation by implementing an orthogonal Least-Square Learning in order to design a basis function (Wang and Mendel, 1992c). By using SOM, Nie et al. (Nie et al., 1993) have shown that the factorised basis functions in RBF are equivalent to the membership functions in the fuzzy system. The combination of both techniques,

ANN and FS is expected to improve the overall performance of an adaptive filter system which is based on the Fuzzy Neural system.

4.1.1 Formal description of Fuzzy-CPN

The fuzzy neurons emit bounded signals from some minimum value to maximum value i.e. [0,1]. At each instant, the n-vector of a neuron output defines a fuzzy unit or **fit vector**. Each fit value indicates the degree to which the neuron or element belongs to the n-dimensional fuzzy set. The state space of the fuzzy neuron that is the set of all possible fuzzy neuron outputs is equal to the set of all n-dimensional fit vectors. That state space are equal the to unit hypercube $I^n=[0,1]^n=[0,1] \times \ldots \times [0,1]$, which is the set of all vectors of length n with co-ordinates in the unit interval [0,1].

In this proposed model, the set of processing units, state of activation and the output of each processing unit are described in this section. The processing units of this proposed model are fuzzy neurons, defuzzifier neurons and defuzzifier control neuron. These neuron models which represent the processing unit and which build the Fuzzy-CPN model are defined in this section. In this model, a fuzzy neuron has some characteristics :

- The process in a fuzzy neuron is the fuzzy process (Lee and Lee, 1975) F in Eq 3.1 including fuzzification or fuzzy inference.
- The output signal is bounded i.e. [0,1] and the value of the y in Eq 3.3 is [0,1].
- The output of a fuzzy neuron is interpreted as the membership value or fit value not as the activation value. Therefore, q in Eq. 3.1 is not interpreted as the final output of a neuron.
- There is no inhibitory input to the neuron.

DEFINITION 4.1

A formal fuzzy neuron is a triple $FN = \{\mathbf{w}, \rho, \Omega\}$, where :

- $\mathbf{w} = [w_1, \ldots, w_n] \in R^n$ is a weight vector of a neuron.
- $\rho = \{\rho_1, \ldots, \rho_v\} \in R$ is a set of parameters of the membership function.
- $\Omega : R \rightarrow R$ is the fuzzy membership function.

A fuzzy neuron is depicted in Fig. 4.1.

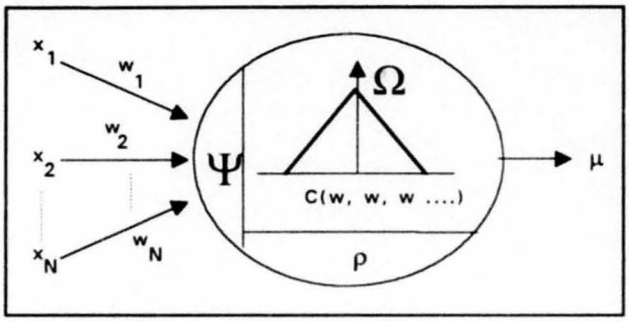

Figure 4.1 Fuzzy-neuron (FN).

As shown in Figure 4.1, a fuzzy neuron operates on N-tuples of input $\mathbf{x} = [x_1, \ldots, x_N] \in R$ and produces a real scalar $\mu = [0,1]$ as an output. The weight vector **w**, encodes the centre of receptive field $C(w_1, w_2, \ldots, w_N)$ of this fuzzy neuron. The spread of the receptive field is determined by a parameter r. These values, **w** and r are adjusted during the training phase.

The output value μ of the fuzzy neuron is defined as follow:

$$\mu = \Omega(d|\rho) \tag{4.1}$$

where the function Ω is the membership function with ρ as the set of the parameters of the function Ω and $\rho = \{r\}$. d is a similarity distance measure :

$$d = \Psi(\mathbf{x}|\mathbf{w}) \tag{4.2}$$

The value μ, of the function Ω is not interpreted as the activation value, but as the membership value or the fit value of the input vector. The function Ψ is the similarity function between **x** and w. This function is a distance measure function such as Euclidean Distance or Absolute Value Distance (Zhou, 1988). In this work, the Euclidean distance is used as the similarity function. Therefore, Eq. 4.1 and Eq. 4.2 can be written in the following equations:

$$\mu(d) = \left(1 + \left(\frac{d}{(r-d)}\right)\right)^{-1} \tag{4.3}$$

where the function Ψ(.) is :

$$\Psi(\mathbf{x}) = \sqrt{\sum_{j=1}^{n} (x_j - w_j)^2} \tag{4.4}$$

where r is the parameter of the membership function. In this model, a triangular membership function is used, and r represents the farthest deviation of the membership function. This membership function has been shown in Figure 3.11. The Eq. 4.4 shows the basis function of this fuzzy neuron which is a radial basis function form, therefore this fuzzy neural model behaves likes RBF-Net which is stated in Eq. 3.9. However, the difference is that the membership function used in this proposed model is a triangular shape and most of the RBF-Nets use Gaussian function to build the neurons as shown in Eq. 3.9 and the learning paradigms.

This fuzzy neuron model is different from the fuzzy-neuron which has been developed by Lee and Lee (Lee and Lee, 1975). The fuzzy-neuron in this model works as a rule and partitions the input space. This neuron will be sensitive to a particular receptive field of the input space. The centre of the receptive field is

determined by the weight vector w, and the membership function parameters determine the response of the neuron to the receptive field around the vector w.

A model for designing a membership function for a fuzzy neuron using example based learning has been developed by Yamakawa and Furukawa (Yamakawa and Furukawa, 1992). The partition of the input and output space can be achieved by employing an adaptive clustering technique performed by the ANN model. The result of this clustering task is used as the parameters which build the membership function of the fuzzy system.

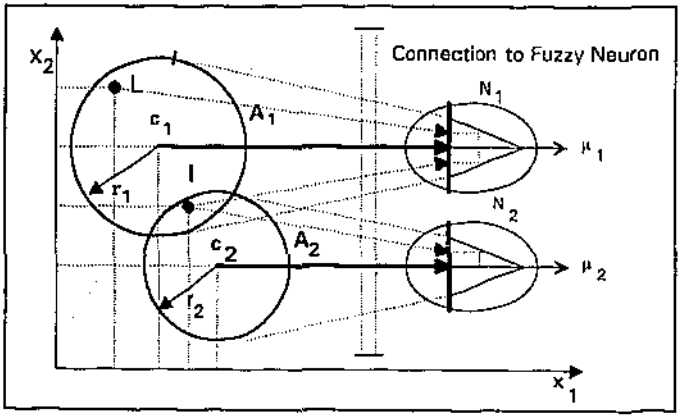

Figure 4.2 The relation of the input space and output of the fuzzy neuron

Figure 4.2 is used to give the explanation of the mechanism of the fuzzy neuron. In Figure 4.2 the input space is a 2-dimensional space (x_1, x_2). The area A_1 is the receptive field of the neuron N_1 and the area A_2 is that of neuron N_2. These receptive fields are determined by r_1 and c_1 for N_1, and r_2 and c_2 for N_2.

Let $I(I_1, I_2)$ be the input vector. It falls into the overlap region of both receptive fields of N_1 and N_2. The fuzzy neuron N_1 produces μ_1 as the output and fuzzy neuron N_2 produces μ_2. Both μ_1 and μ_2 are greater than zero, because the input

vector falls into their receptive fields. $L(L_1, L_2)$ is another input vector and it falls the N_1. Therefore, the value of μ_1 is greater than 0 and μ_2 is equal to zero. It means that only fuzzy neuron N_1 is active. The output values of the fuzzy neurons, μ_1 and μ_2 depend on the position of the input vector in the receptive fields of the fuzzy neurons.

During the training phase, only the parameters of fuzzy neuron which produces the highest output will be adjusted. However, in the recall phase all fuzzy neuron outputs are used to determine the final output by applying a proper defuzzication technique in order to combine the contribution of each fuzzy neuron in determining the final output. The defuzification task is performed by a defuzzifier neuron. The defuzzifier neuron is defined as follow :

DEFINITION 4.2

A defuzzifier neuron is formalised by $DN = \{\mathbf{w}, M, G\}$ *where*:

- $\mathbf{w} = [w_1, w_2, \ldots, w_n]$ is the weight vector.
- M is the defuzzification method being used.
- G is the gain control of the neuron.

The defuzzifier neuron operates on n-tuples of input $\mathbf{u} = [u_1, \ldots, u_n] \in \mathbf{R}$ and produces a real scalar y as an output:

$$y = G \cdot M(\mathbf{u}|\mathbf{w}) \tag{4.5}$$

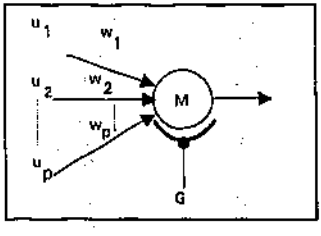

Figure 4.3. Defuzzifier neuron (DN)

In this proposed model the centroid defuzzification is used and each component of the Eq. 4.5 can be written as :

$$M = \sum_{j=1}^{p} u_j w_j \qquad (4.6)$$

The gain control G is controlled by the defuzzifier control node. This neuron has non-adjustable weights.

DEFINITION 4.3

The defuzzifier control neuron is denoted by DCN :

- $w = [w_1, w_2, \ldots, w_n]$ is the weight vector of DCN. These weights are non-adjustable.

The output of this neuron is:

$$G = \frac{1}{\sum_{j=1}^{p} u_j} \qquad (4.7)$$

By combining the ANN technique and FS approach, the proposed model performs tasks including: partitioning the input space and the output space, defining the mapping association between a fuzzy patch in the input space to the associated fuzzy patch in the output space, building the fuzzy membership function of each fuzzy neuron, and obtaining the centroid value of each fuzzy neuron. In the learning process, the model generates the fuzzy neurons the number of which is equal to the number of membership functions. Therefore, it is not necessary to pre-define the number of rules and the membership functions. The number of rules and the membership function parameters will be determined automatically in the learning phase.

4.1.2 Counterpropagation paradigm

In this section, the propagation rules for propagation patterns of activities through the network of connectivities are described. The flows of Transmitting Feedback Vector, Transmitting Feedback Signal, Receiving Feedback Signal and Receiving Feedback Vector in the Feedback Path and Feed Forward Path are described as well.

In the proposed model, the Counterpropagation Network (CPN) architecture is chosen as the basic paradigm because it is simple, fast, and easy to train. CPN has a good representation of the statistical model of the input space and in the network mapping problem CPN has a closed form of the means square error. The capability of CPN to perform fast learning is useful for some applications that need an on-line learning mechanism, e.g. adaptive control, trajectory problem of teach and play robot, adaptive filter.

Counterpropagation networks can be used for pattern classification where template matching and template interpolation are desired (Hecht-Nielsen, 1987). CPN has been applied in many problems such as Dolphin echolocation (Roitblat et al., 1989), digital feedback equaliser (Manabe and Kaneda, 1991). In addition, it has been implemented in a VLSI system (Kwan and Tsang, 1990). It can be used for multidirectional associative memory (Naik et al., 1992) and data compression (Liu et al., 1992).

The CPN architecture is built by combining the Kohonen self-organizing map and the Grossberg Outstar architecture. The Kohonen self-organizing map performs the clustering task and the Grossberg Outstar performs the encoding function. In general, it can be stated that the CPN uses an adaptive lookup table mechanism to perform the mapping while the table is obtained by training.

To describe the operation of a typical CPN, let (x_i, y_i) be one of the input-output pairs of a set of examples of a function ϕ:

$$\phi : R^n \rightarrow R^m, \ where \ y_i = \phi(x_i) \tag{4.8}$$

Assuming that this set of examples cover all the characteristics of the input-output relationship of the function, after training the CPN with this set of examples, the weights of the network are adjusted according to the training set in such a way that the inputs are classified into clusters. The final clusters can be modelled in the form of a lookup table with N entries, where the number N is equal to the number of clusters (Hecht-Nielsen, 1988).

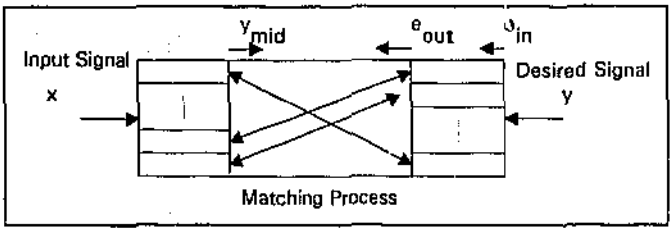

Figure 4.4 Counterpropagation mechanism

Neurons in the CPN model are the Bi-Directional Model (BDM). This model of neurons usually are used for the supervised learning. Learning in CPN is performed in two directions: feed forward path and feedback path. The input pattern flows from the input layer to the middle layer and the desired output pattern flows from the output layer to the middle layer. They match in the middle layer as shown in Figure 4.4. The e_{in}, a feedback signal, flows from the output layer and is used to update the w parameter of the output layer but they do not continue to flow to the lower layer as in the backpropagation model. The e_{out} of the output layer which is the error signal that flows to the lower layer is used in the matching process at the middle layer when

the signal y_{mid} of the middle layer arrives. This matching mechanism is suitable to provide a basic framework for a fuzzy neural system, which will be explained later. The recall mechanism to produce the output of a single neuron in the middle layer of CPN can be interpreted as :

IF *the input is in cluster* x_i THEN *the output is* y_i

In order for a CPN to perform mapping of a continuous function with good approximation, a large lookup table is required which means that a network with a large middle layer is required (Hecht-Nielsen, 1987). By splitting a large middle layer to the smaller SOMs (Lin et al., 1989), the number of neurons in the middle layer can be reduced by a medium amount.

As shown by Wang and Mendel (Wang and Mendel, 1992a), a set of fuzzy rules can be used for a universal function approximation. In the proposed architecture, the fuzzy set approach is incorporated into the middle layer of a CPN to provide a smoother mapping. Thus, the output of the middle layer gives a membership value of the input belonging to each cluster. The original Grossberg outstar layer is replaced by a layer of neurons which are controlled by a defuzzifier control neuron. Therefore, the proposed model is identified as a **Fuzzy-CPN** model.

During training, the modified CPN generates the fuzzy neurons from the set of examples of input-output pairs. Each fuzzy neuron represents a fuzzy rule. Each example of input-output pairs is only required to be presented to the input of the network once and no iterative training is required.

In recall phase, the proposed network architecture does not produce a cluster as a winner in the middle layer, but produces the membership values of the input with

respect to each class. By using proper defuzzification method, a continuous function can be obtained at the output layer. Moreover, this mechanism is suitable for on-line learning, because the system does not require iterative learning. Especially in a time series prediction problem, after performing a prediction, the actual value of the next time series can be used to refine the network for predicting the future time series.

This method is different from the interpolative mode of conventional CPN (Hecht-Nielsen, 1988c). In the interpolative mode, more than one node in the middle layer can win and the winning nodes are weighted with a fraction number and the sum of all weighted number is equal to one. Therefore, there are the first, the second, and the third winners. To perform the interpolative mode, a priori knowledge about the problem is required to define these fraction numbers.

In the proposed network, by the use of fuzzy logic, the number of neurons in the middle layer is greatly reduced which leads to shorter training time, and the network has the capability to perform function approximation without the requirement of a priori knowledge. Similar work in combining fuzzy logic and the neural system have been reported using different approaches such as Fuzzy ARTMAP (Carpenter et al., 1992) and Fuzzy Min-Max (Simpson, 1993). Some applications of these hybrid models can be found in the speech recognition (Jianxin et. al., 1992).

4.1.3 Architecture of Fuzzy-CPN

This section describes the pattern of connectivity among processing units in the Fuzzy-CPN. Since the number of fuzzy neurons as the processing units and the membership functions are not pre-defined, there should be a learning mechanism

which can add and delete fuzzy neurons automatically. It leads to the concepts of self growing and self-organising in the neural network architecture, which has capability to grow and prune the fuzzy neurons and build the network structure while training is performed (Fritzke, 1993a). This class of networks has more capability for quantising the input space compared to Kohonen approach (Fritzke, 1993b). Some self-growing network architectures have been developed, for example, CombNET-II (Iwata et al., 1992), Dystal (Dynamically Stable Associative Learning) (Barbour, et al., 1992), DIGNET (Thomopoulos, 1991) and by combining with Genetic techniques which has been developed by Nolfi et al. (Nolfi et al., 1992),

The proposed network uses fuzzy leader clustering technique instead of Kohonen layer for the middle layer, and Grossberg Outstar is replaced by the defuzzifier neurons to store the centroid value of each class. By adding a defuzzifier neurons, the defuzzification is based on the centroids obtained in the output layer. The architecture of the network is shown in Figure. 4.5.

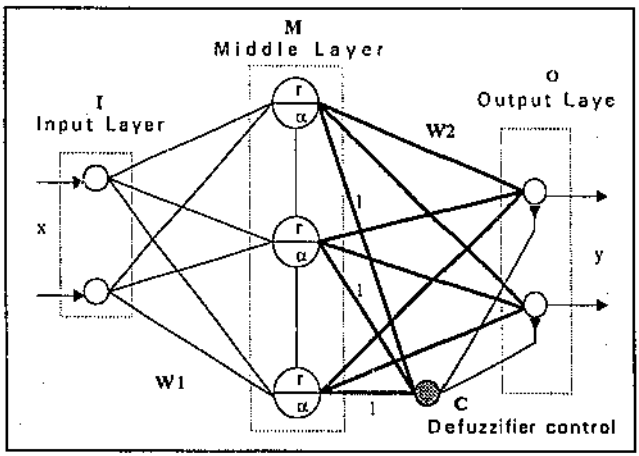

Figure 4.5 Fuzzy Counterpropagation Network architecture

By using Eq. 3.13, the Fuzzy-CPN can be formalised as follow :

$$G_{Fuzzy\text{-}CPN} = \{I_1, .., I_n, M_1, ..., M_l, O_1, ..., O_k, C\} \quad (4.9)$$

where I is the input layer for the n-dimension input space, M is the middle layer with the maximum number of neurons equal to l, O is the output layer for the k-dimension output space and C is the special control neuron. Then each component of G in Eq 3.14 are defined as:

$$I = [IP, R_{IP\text{-}I}, T_{I\text{-}M}] \quad (4.10)$$

$$M = [FN, R_{I\text{-}M}, \{T_{M\text{-}O}, T_{M\text{-}C}\}] \quad (4.11)$$

$$O = [DFN(R_{O^G\text{-}C}), R_{O\text{-}M}, T_{O\text{-}OP}] \quad (4.12)$$

$$C = [DCN, R_{M\text{-}C}, T_{C\text{-}O^G}] \quad (4.13)$$

where the subscripts for the T and R represent the direction of the connection between the component indicated by the subscripts. The model $G_{Fuzzy\text{-}CPN}$ consists of three layers of neurons, I, M and O in the Eq. 4.9 and C is the special neuron model as defined in Eq. 4.13. The first layer I consists of linear processing units IP which receive the input vector $R_{IP\text{-}I}$ and the number of neurons in the input layer depends on the dimension of the input space n. The second layer or the middle layer is a set of fuzzy neurons FN, which are defined in Definition 4.1. The middle layer is a self-growing structure. During the training phase, it starts with no fuzzy neuron, and will grow depending on the training patterns and the final number of neuron in the middle layer is denoted by l. The input layer and the middle layer are fully connected by $\mathbf{W1} = \{\mathbf{T}_{I\text{-}M}, \mathbf{R}_{M\text{-}I}\}$. Therefore, $T_{I\text{-}M}$ is equal to $R_{I\text{-}M}$ for each component in I and M.

The third layer is the output layer which consists of the defuzzifier neuron $DFN(.)$ defined in Definition 4.2. The number of the neurons in the output layer is

determined by the dimension of the output space, denoted by k. The gain of the defuzzifier neuron is controlled by the defuzzifier control neuron, *DCN* which is defined in Definition 4.3. The middle layer and output layer are both fully connected by $\mathbf{W2} = \{\mathbf{T}_{M\text{-}O}, \mathbf{R}_{O\text{-}M}\}$ and non adjustable weights are used to connect the middle layer to the defuzzifier control neuron which is $\{\mathbf{T}_{M\text{-}C}, \mathbf{R}_{C\text{-}M}\}$. These non-adjustable weights are set to be equal to 1.

W1 is the synaptic weight matrix which stores the information of the centre of the receptive fields c of each neuron in middle layer. Furthermore, each fuzzy neuron at middle layer stores the value of farthest deviation of its receptive fields r and its learning parameter α. Each fuzzy neuron at the middle layer does not have the same learning parameter. **W2** is the synaptic weight matrix which encodes the information about the output prototype of each fuzzy neuron. It will be used in the defuzzification step for calculating the final output.

The first layer and the middle layer act as the clustering and membership assignment of the input vector. The output layer works as the defuzzification mechanism to defuzzify the fuzzy output value of each fuzzy neuron in order to produce the final output.

Clustering and Membership Layer

A clustering network performs the clustering process by comparing the input pattern with the templates which are stored as the weights of the network. Some neural network models can be used as a pattern classifier or for data clustering. Self-Organizing Map (SOM) (Kohonen, 1990) is one such model and Adaptive Resonance Theory (ART) (Carpenter et al., 1988) is another model. SOM using Fuzziness Measures (Ghosh et al., 1993) has been applied for extracting an object

from noisy image. The Kohonen clustering technique has several problems which have been addressed by Bezdek et al. (Bezdek et al., 1992). The final output usually depends on the order of the input pattern sequence and termination is not based on optimising a particular cost function. Therefore, different initial conditions produce different results. These problems is solved by applying Fuzzy Kohonen Clustering Network. The ART is another unsupervised neural network which can deal with plasticity-stability dilemma, but will give bad performance when the input patterns are coloured by the noise (Lippman, 1987).

In the ART or SOM clustering strategy, the distance between the input vector and each template is used as the value that judges the winning cluster to perform the clustering process. In the proposed model the membership value assigned by the fuzzy neuron is used to perform the clustering in determination of the winning template. However, it is different from the Adaptive Fuzzy Leader Clustering proposed by Newton et al., (Newton et al., 1992) that used ART to perform the recognition of the input pattern for the clustering of arbitrary data patterns.

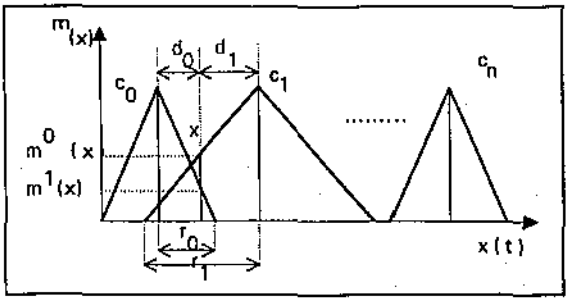

Figure 4.6 Membership function

The classification and membership layer, which is the combination of the input and middle layers, partitions the input space into clusters. Given an input vector **x** as

shown in Figure. 4.6, the Euclidean distances between the input vector and all prototype vectors are calculated e.g. d_1, and d_2. By the Euclidean distances, the membership value for each class can be obtained e.g. $m^0(\mathbf{x})$ and $m^1(\mathbf{x})$. In normal operation, the membership value is then used to derive the output vector.

In the training process, the weight vector between a node in the input layer and that in the middle layer, **WI** (as shown in Figure. 4.5) is adjusted to capture the prototype of each class which can be used for the membership value decision. Only weights which connect the closest cluster (highest membership value) with the input vector will be adjusted. This middle layer works as a matching process, i.e. each cluster of input space has an associative output value , or it can be stated as :

IF *the input vector is in Receptive Field of fuzzy neuron n* THEN *Output is* y_n

It is similar to the rule in fuzzy logic term. All of those rules or matching process of a fuzzy neuron are built up in the training phase automatically and they are not pre-defined. This mechanism provides the network a capability to learn the input-output pairs and to build the fuzzy rules according to the input-output relationship.

Similar techniques to the proposed model have been developed such as Fuzzy Elastic Clustering (Srikanth, 1993), Fuzzy C Spherical Cells (Krishnapuram et al., 1992), Nearest Neighbour Pattern Classification method (Bang and Park ,1991). The problem of those techniques are that the ANN needs a sufficiently large amount of training. In the proposed model the clustering process is performed by using minimal iterations.

Since the optimal capacity of the network is affected by the number of clusters (Turunini, 1990), some mechanism must be provided to achieve the optimal capability of the network to store the templates. With pruning and adding neurons and clustering in product space (Berenji, et al., 1993), the optimal capacity of ANN can be improved. This strategy will be implemented in the learning algorithm.

Defuzzifier neuron and output layer

The neuron at the output layer of this proposed model does not just perform the summation over the input signal. Since the matching process at the middle layer does not produce only one winner as the most matched fuzzy neuron, the output layer has to provide a mechanism to combine the outputs of the fuzzy neurons from the middle layer. This combining process has to represent the degree of matching of each rule. It is similar to the defuzzification process.

The defuzzification process is done by the defuzzifier neuron defined in Definition 4.2 and defuzzifier control neuron defined in Definition 4.3. The defuzzification method used is the centroid method or the centre of gravity. All weights which connect the defuzzifier control neurons (DCN) to the fuzzy neuron in the middle layer have values equal to 1. Therefore, the output of the defuzzifier control neuron is the total of the membership values to each prototype. The neuron in the output layer which is defuzzifier neuron is rather different from conventional neurons, due to the adaptive gain characteristic of this neuron. The gain of this defuzzifier neuron is controlled by the defuzzifier control neuron.

4.2 Fuzzy-CPN Mechanism

The Fuzzy-CPN has different behaviour and learning algorithm compared to the conventional CPNs. The similarities are only the basic paradigm, and the way of the networks feed the training signals and the propagation of the signals in the networks. This proposed network model is one of the k-winner take-all network. The equilibrium state of this class of ANNs has been addressed by Majani et al. (Majani et al., 1992). The middle layer has self-growing and self-organizing behaviours and it is addressed in the following section followed by the learning rule and recall algorithm of the Fuzzy-CPN. Patterns of connectivity and the environment within which the system must operate are explained in this section.

4.2.1 Adaptation mechanism

The complexity of learning is dictated by the number of middle neurons. Thus, the number of middle neurons should be kept as small as possible to ensure low complexity of learning. A network information criterion (Murata et al., 1992) can be used to determine the number of middle neurons in a multi-layer perceptron neural network. The minimisation of an information criterion has been applied as well for a competitive and self-organising network (Benaim, 1991).

From the implementation point of view, the resource of computation is limited, i.e. the number of fuzzy neurons in the middle layer is limited. In order to achieve an overall good performance of the network model any unnecessary component must be removed from the system without decreasing the performance. Therefore, a mechanism to optimise the use of the fuzzy neurons at the middle layer is introduced. In this proposed model, this mechanism is achieved by constructing the middle layer

as a self-growing structure of fuzzy neurons. Each fuzzy neuron is generated when it is required and will be annihilated automatically when it is not used efficiently with respect to the training patterns. During training phase, the Fuzzy-CPN performs the adaptation process in order to arrange the structure to tackle the problem. In most ANN models, the component of Eq 3.13 and 3.14 are fixed. In this model, at the initial time of the training, the number of fuzzy neuron at middle layer is not determined. It grows during the training. This means that the component in Eq. 4.9 will be changed during the training process. Therefore, this architecture can be classified as Structure Level Adaptation Neural Network (Lee, 1922). During the training process, fuzzy neurons will be generated, until all available source for creating fuzzy neurons has been used. A fuzzy neuron which has not been used for a fixed time will be annihilated, and then its resource can be used again. Since the structure of the network always adapts during the training process, there are three main activities of the neurons at the competitive layer. They are neuron generation, neuron adaptation and neuron annihilation.

Neuron generation

A fuzzy neuron in the middle layer is generated when the network cannot classify the current input vector according to the pre-defined criteria and there is still available resource to generate a fuzzy neuron. Since the memory is limited, there is a circumstances when it is not possible to generate a fuzzy neuron, i.e. all memory that is reserved for the fuzzy neurons have been used.

If the model has enough fuzzy neurons to process the clustering of input space, then during the training phase the parameter structure of each fuzzy neuron will converge to a small neighbourhood region around a certain value in the neuron

parameter space. If the neural network does not contain enough fuzzy neurons to learn the clustering of problems the parameters will oscillate and a new fuzzy neuron should be generated. When a fuzzy neuron is generated, then it must be placed to the position where it is most needed. To achieve this requirement in this model, the current input pattern is used as the centre of the receptive field of the fuzzy neuron. Therefore, it is assumed that the first pattern which causes the generation of a fuzzy neuron acts as the leader of clustering. The neuron generation adds the component M in the set $G_{\text{Fuzzy-CPN}}$ in Eq. 4.9. By generating a fuzzy neuron means that adding a fuzzy rule in the fuzzy rule base.

Neuron adaptation

For a fuzzy neuron, the adaptation is the adjustment of **W1** and the r value of the neuron. It is performed only to the winning fuzzy neuron in the competition during training phase. This mechanism adjusts the receptive field of the winning fuzzy neuron in order to capture better representation of the input vector. Basically, this adjustment is performed to tune the parameter of the membership function in the fuzzy term.

The adjustment of **W1** and r are similar to the changes of the IF-part of a fuzzy rule of a fuzzy system. The adaptation of the defuzzifier neuron is performed to **W2** of the architecture. Since the value of **W2** represents the consequence part of the rule, this adjustment tunes the THEN-part of the fuzzy rule of the system. It is clear that the neuron adaptation provides a mechanism to tune the membership function and the fuzzy rule itself. This adaptation has to be maintained so as not to oscillate, and this is achieved by using a decay function that control the adaptation gain.

Neuron annihilation

A fuzzy neuron will keep finding the position in the fuzzy neuron parameter space in this clustering layer which is a competitive layer. If a fuzzy neuron does not form the correct interconnection among other neurons, then it will die in the early development. The neurons compete with each other for resources and each neuron tries to inhibit other neurons from taking exactly the same functional role that it plays in the network. This behaviour is used to optimise the network. The annihilation of neuron deletes the component M in the $G_{Fuzzy\text{-}CPN}$ in Eq. 4.9. However, the forgetting mechanism that provides the ability to delete specified equilibrium points from a given set stored equilibrium has to be performed without affecting the equilibria in the given network (Yen, 1991). Therefore, to annihilate a fuzzy neuron, some criteria have to be satisfied. Firstly, the fuzzy neuron should not be a functioning element in the network. It means that by deleting that fuzzy neuron, the performance of the network will not be degraded Secondly, the fuzzy neuron is a redundant in the network. It means there are two or more fuzzy neurons that produce a similar output for the same input.

The first criteria can be checked by watching the activity of a neuron. If the output activity is fixed over a very long time, i.e., the cluster never wins, so not contribute to the function of the network, because it does not generate any information in its output. To measure the output activity, the Activity Variance (AV) can be used (Lee, 1992). The Activity Variance is related to the information content of the output signal of a neuron The entropy of the neuron can be used to describe the activity variance. If this value is zero it means that no information is generated by this neuron and it does not perform any signal processing function. Therefore, it can

be removed from the network. A neuron which is never used or inefficiently used can be evaluated using the entropy of the output of that neuron. If the entropy is less than a minimum level, then the neuron will be annihilated.

For the second criterion, it can be checked by watching the dependency between output of the fuzzy neurons. If two neurons are totally dependent, then one of the neurons can be annihilated without affecting the performance of the network. If the weight vectors of two neurons are very close to each other then one of them can be eliminated.

Hence, to optimise the middle neuron without violating those two criteria, the penalty technique is applied. In applying the penalty technique, some cost function can be used. According to Matsuyama (Matsuyama, 1992) there are many cost functions such as competitive learning with conscience and entropy minimisation, which have been applied as the penalty measurement for the competitive mechanism. The amount of information gained during the training phase is one of the important characterisations of learning process. Measuring this information gain can be obtained by the statistical distance between pre and post training (Lebin et al., 1990). Since a middle neuron acts as a rule, the minimum entropy can be used to find the information gain from the training examples. This method has been used for rule learning from examples in the area of knowledge base systems (Pitas et. al., 1992).

In this model, for simplicity of implementation, frequency occurrence which determines the entropy is used. Therefore, the learning parameter will be adjusted differently with respect to the cost function for each neuron which involves in the competitive learning mechanism.

4.2.2 Learning mechanism

In this section, each component of the Parameter Adaptation Automata as stated in Eq. 3.10 for the neurons in the Fuzzy-CPN is to be specified. The proposed model applies the learning mechanism which is a model of the constructive learning by specialisation (Refenes, 1991). A fuzzy neuron specialises to a particular receptive field in the input space to produce the associate output value. The learning mechanism of this ANN model is pictured in Figure. 4.7.

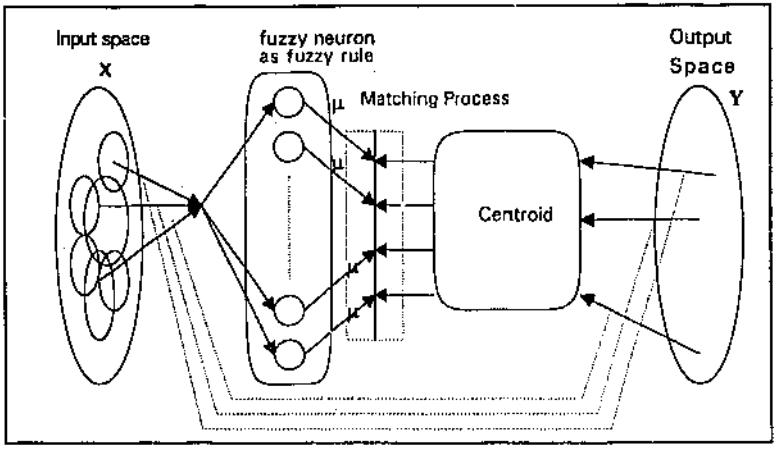

Figure. 4.7 Learning mechanism in Fuzzy-CPN model

In Figure 4.7, the learning mechanism controls the generation of each fuzzy neuron from the numerical input-output data (X,Y). Each fuzzy neuron acts as a fuzzy rule. On every representation of training pattern (X,Y), which is represented by a dashed line in Figure 4.7, the learning mechanism automatically partitions the input space and generates the membership function for each data without any a priori knowledge. It means that at this step, the IF-part of the fuzzy rule is being built. By performing a matching process with the output vector, the learning mechanism builds THEN-part

of the fuzzy rule. The matching process of the fuzzy neuron is performed according to the μ of each fuzzy neuron.

Therefore, a fuzzy neuron at the middle layer stores the information about the IF-part which is the mean of the cluster and the farthest deviation of the cluster. The boundary of the cluster has the membership value equal to zero and the centre of the cluster has the membership value equal to one. The THEN-part is encoded in the **W2** as the centroid values of each fuzzy rule's output, i.e. the output value of each fuzzy neuron.

This model has some important parameters which are used to represent the conditions and relations of a pattern in the storing mechanism.

Defines

r_i the maximum deviation of i^{th} cluster from the centre of i^{th} cluster.

c_{ji} the prototype vector of i^{th} cluster of j dimensions.

X_j the input vector of j dimensions.

Y_k the target vector of k dimensions.

The learning mechanism can be formulated as follows :

Step 1. Start with 0 number of fuzzy neurons.

Step 2. New input vector X_j is applied to the input layer.

Step 3. The output of each fuzzy neuron at middle layer is calculated.

Step 3a. The Euclidean distance d_i's from the input vector to all prototype vectors, each of which is stored in a fuzzy neuron in the middle layer that exist are calculated.

$$d_i = \sqrt{\sum_{j=1}^{n} (c_{ji} - X_j)^2} \qquad (4.14)$$

Step 3b. Each fuzzy neuron in the middle layer calculates the membership value of the input associated with the cluster, which are represented by fuzzy neurons that has been created. For ease of implementation, the triangular shape for the membership function, μ_i is used. This function is rather similar to the membership function proposed by fuzzy MLP (Pal, et. al. 1992)

$$\mu_i = \left(1 + \left(\frac{d_i}{(r_i - d_i)}\right)\right)^{-1} \tag{4.15}$$

Step 4. The fuzzy neuron which produces the highest membership value is determined. It is indexed as *win*. At this step the competition mechanism is executed.

Step 5. The Parameter Adjusting Function δ in Eq. 3.11 for this model is applied in this step. For this model, the counterpropagation paradigm is applied, therefore, the weight vectors from both side **W1** and **W2** that connect to the fuzzy neuron which produces the highest output is updated, according to :

$$W1_{win}^{new} = W1_{win}^{old} + \beta\left(X - W1_{win}^{old}\right)$$

$$W2_{win}^{new} = W2_{win}^{old} + \beta\left(Y - W2_{win}^{old}\right) \tag{4.16}$$

where β is the learning parameter in the general. This value is updated automatically for each iteration.

$$\beta = \frac{\alpha}{\alpha + 1} \tag{4.17}$$

where α is the Parameter Control State of each fuzzy neuron, which has the initial value equal to 1.0. In this model, to update the value of α for each

training, the Parameter State Transition Function σ in Eq. 3.12 is defined in order to calculate α iteratively by :

$$\alpha_{win}^{new} = \left(\frac{1}{\alpha_{win}^{old}} + 1 \right)^{-1} \tag{4.18}$$

In this case, the farthest deviation for the winning fuzzy neuron is updated according to :

$$r_{win}^{new} = r_{win}^{old} + \sqrt{\sum_{j=1}^{n} \left(W1_{j,win}^{new} - W1_{j,win}^{old} \right)^2} \tag{4.19}$$

This step is the neuron adaptation mechanism. The fuzzy neuron adjusts its receptive field which is determined by the values of the farthest deviation r, and the centre of receptive field which is encoded in the **W1**. The response of the fuzzy neuron to future learning, α, is updated as well. Therefore, if a neuron has learnt more, the neuron will be less responsive to the learning, and will adapt with a very small change. The β value always decays. Thus it guarantees that the updating of the weight vector will not oscillate and will converge to a final value. This adaptation process is shown in Figure 4.8 by using 2 dimensional input space in x_1 and x_2.

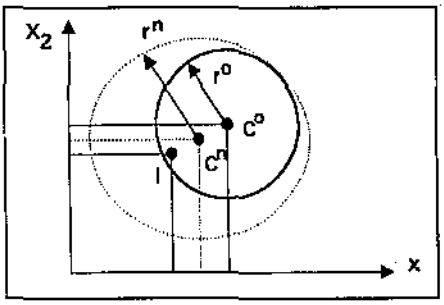

Figure 4.8 The receptive field adaptation of a fuzzy neuron

Let I be the input vector, in Figure 4.8, as shown by a dot near the boundary of the circle. C^o is the centre of the cluster before the adaptation and C^n is the centre of cluster after the adaptation. r^o is the farthest deviation of the cluster before adaptation and the r^n is the farthest deviation of cluster after the adaptation. Assuming that the fuzzy neuron has just been generated, thus α is equal to 1. This fuzzy neurons wins the competition. Therefore, this fuzzy neuron has to be updated. Since α is equal to 1, C^n is just the midpoint between C^o and I. However, the farthest deviation of cluster, r^o has to be updated. In updating the farthest deviation of cluster, the consistency between the area which is covered by the previous parameters of fuzzy neuron and the new parameter should be maintained. Therefore, the previous area of cluster has to lie in the area of the new cluster. It is shown by the circle with the dash-line. Using this approach, if a fuzzy neuron is activated by an input vector, after updating the parameter of that fuzzy neuron, the same input vector will still be able to activate the same fuzzy neuron. The difference is only the membership value which is produced by that fuzzy neuron with respect to the same input vector.

Step 6. When executing Step 3, if all fuzzy neurons produce zero as the output, a new fuzzy neuron will be generated and the input pattern is used as the centre of the receptive field of the fuzzy neuron. In this case, the farthest deviation is set to a default value, ρ.

$$W1^{new} = X; \quad W2^{new} = Y; \; r^{new} = \rho \; ; \qquad \alpha^{new} = 1 \qquad (4.20)$$

This step is the neuron generation process. The learning parameter ρ determines when a fuzzy neuron should be generated in the learning process. It is similar to the vigilance parameter in the ART model (Carpenter et al., 1986). Using a smaller value of this learning parameter, the network tends to produce more fuzzy neurons, i.e. it tends to generate a new fuzzy neuron, when an input vector is applied.

Step 7. When Step 6 cannot be accomplished because of all resources that are required to create a fuzzy neuron have been used, i.e. there is not enough memory to create a new fuzzy neuron. The fuzzy neuron which has the highest α value and exceeds a pre-defined threshold, α_{limit} is chosen. And the parameter of this fuzzy neuron is set to be :

$$W1^{ann} = X \; ; \quad W2^{ann} = Y \quad ; \quad r^{ann} = \rho \quad ; \; \alpha = 1 \qquad (4.21)$$

This step is neuron annihilation process. A fuzzy neuron which is not used so much should be annihilated in order to decrease the learning complexity. It is performed by finding the fuzzy neuron which has the highest value of α. The higher value of α means that that fuzzy neuron has not won very often. Therefore, it can be deleted because it does not contribute so much in the competition process in the training mechanism. However, a threshold for α has to be pre-defined in order for the annihilation process to be executed to the fuzzy neurons which have the α value more than the threshold.

Step 8. In the situation when Step 7 cannot be performed, i.e., there is no fuzzy neuron which has the α value more than the threshold. A fuzzy neuron is chosen by finding the minimum value of S_i of each fuzzy neuron to the input vector.

$$S_i = \left(\sqrt{\sum_{j=1}^{n} (W1_{j,i} - X_j)^2} \right) - r_i \qquad (4.22)$$

These distance are shown in Figure 4.9. The distance S is the distance to the Shell of the fuzzy neuron. This similar distance calculation is applied by the Fuzzy c-Shells Clustering techniques (Dave and Bhaswara, 1992).

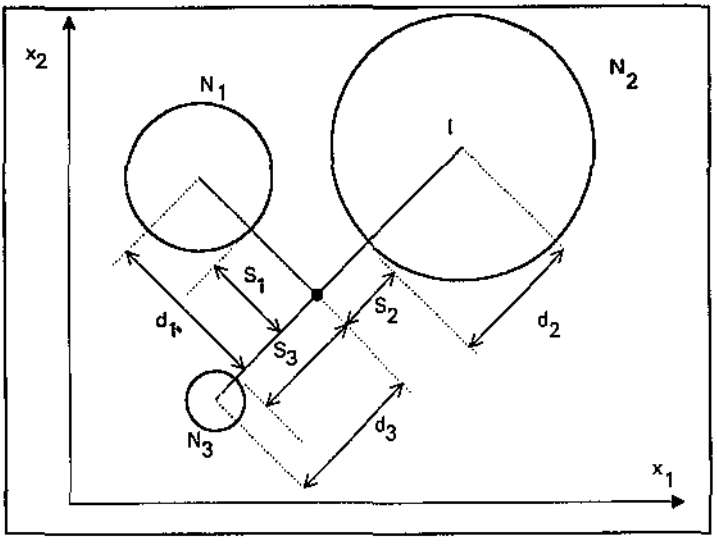

Figure 4.9. The shell distance S

Step 9. Step 2 through Step 8 are executed for all other training patterns.

After finishing training the fuzzy neurons at the middle layer, the input space is covered by clusters as shown in Figure 4.10. There are two main learning parameters which determine the final result they are ρ and threshold of α or α_{limit}. It has to be defined in the beginning of training. However, by setting ρ equal to 0 and the limit of α equal to 1, i.e. there is no neuron annihilation performed, learning can still be executed and the result is the conventional CPN.

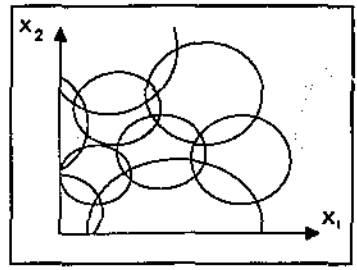

Figure 4.10. Training result of a 2-dimensional input space

Figure 4.10 shows a typical final result after a Fuzzy-CPN has been trained through all training patterns which are assumed to cover the input space. In Figure 4.10 each circle represents the receptive field of each fuzzy neuron at the input space. There are many overlap areas of the receptive fields of the fuzzy neurons. However, during the recall mechanism only one neuron is activated which is a neuron with the highest activation value as in the conventional competitive model. All fuzzy neurons which have the receptive field covering the input vector are activated in different degrees. The defuzzifier neurons determine the final result. It makes this Fuzzy-CPN model different from the other competitive models.

4.2.3 Recall mechanism

This section describes the recall mechanism of this Fuzzy-CPN. The recall mechanism can be implemented in parallel processing, because in the middle layer there is no requirement to find the winning neuron like in the other competitive model. Each fuzzy-neuron processes the input vector and produces the output as the membership value to the defuzzifier neuron without being aware of the activation values of the other fuzzy neuron. It is different from the conventional CPN or any other ANN models with competitive paradigm, which produce the output by finding

the maximum output. Therefore, it reduces the complexity of computation in the middle layer when the recall phase is executed.

The recall mechanism is shown in Figure 4.11.

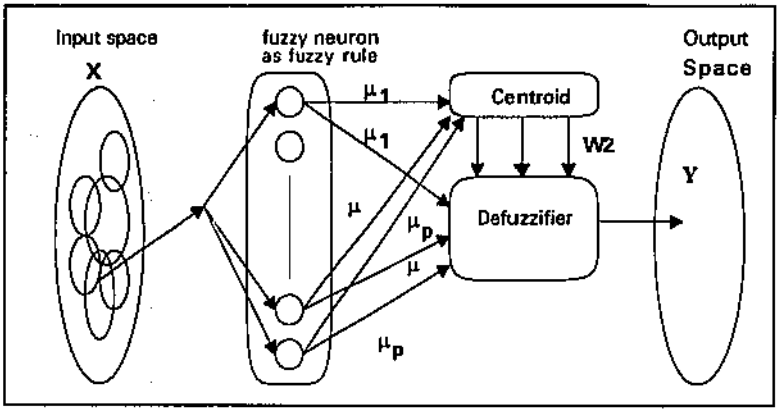

Figure 4.11 Recall mechanism

As shown in Figure 4.11, while the Fuzzy-CPN is performing recall operation, an input vector is given. Then, the middle layer calculates its membership values to each class using Eq. 4.9 and Eq. 4.10. These values (μ_1, μ_1, ... , μ_p) flow into the defuzzifier control neuron and the output layer in parallel. At the same time the centroids values are supplied to the defuzzifier neuron. By using the centroids which are encoded into the **W2**, the output layer calculates the final output value by activating the defuzzifier control neuron. All output values of the fuzzy neuron flow into the defuzzifier control neuron as well.

In producing the output of the fuzzy neuron, the output is not superimposed like in the conventional ANN model. The output has to be averaged with respect to all the rules. The natural alternative is the fuzzy centroid defuzzification. It directly computes the real value output as normalised convex combination of fit values. At

the final stage, the defuzzifier neuron is used to quantise the output. The output of the system has to be able to represent the presence of all rules in the rule base.

After obtaining the output of the middle layer, the final output of defuzzifier control neuron is :

$$Out_{DCN} = \sum_{i=1}^{p} \mu_i \tag{4.23}$$

where p is the number of fuzzy neurons, μ is the input vector to the defuzzifier control neuron. Thus the output from the output layer is equal to:

$$Out_k = G\left(\sum_{i=1}^{p} (\mu_i W2_{ik})\right) \tag{4.24}$$

where G is the gain factor which corresponds to the output of the defuzzifier control neuron:

$$G = \frac{1}{Out_{DCN}} \tag{4.25}$$

Hence the output of the k^{th} output node is :

$$Out_k = \frac{\sum_{i=1}^{p} (\mu_i W2_{ik})}{\sum_{i=1}^{p} \mu_i} \tag{4.26}$$

This recall mechanism is rather similar to the output calculation which is performed by the fuzzy system. The difference is that the inference process and the fuzzification are performed automatically together in the fuzzy neurons and the process of generating and tuning the membership functions are performed automatically by feeding the network with the input-output patterns.

4.3 Function Approximation in Adaptive Filter

Although an ANN has capability for function approximation, there are still many problems in implementing the ANN model for this task. The main problem of the mapping function is the condition when the network underfits or overfits the training data. There is always a trade-off between the generalisation and the complexity of the network (Poggio et al., 1990). Generalisation is not possible if the mapping is completely random (Mostafa, 1989).

Basically, an adaptive system performs adaptation in two steps (Currie, 1992). The first step is the prediction of the output value (a priori output) with respect to the input value and the current state of the system, including static and dynamic states. After that a particular error correction algorithm is applied with respect to the desired signal according to a quality criterion in order to minimise the error between the a priori output and the desired signal. A prediction of the a priori output is performed by applying a function approximation of the input-output relationship. Therefore, a function approximation capability of the adaptive processing should be considered as the main aspect in designing an adaptive system. Since a better prediction mechanism will yield an a priori output closer to the desired signal, a smaller error is produced. Hence, it makes the adaptive system converge faster and has a lower convergence error.

Let f be the transfer function of the system that will be modelled by using an adaptive system. Function f can be the transfer function that performs mapping from the corrupted signal space to the noise-free signal space in the signal correction problem, or from the input space to the output space of an unknown system for system identification. The simplest structure is a time-domain adaptive filter with

FIR structure. The output value $y(t)$ is a linear combination of the input and the delay input signals $[x(t), x(t\text{-}1), \ldots, (x(t\text{-}k)]$:

$$f: (x(t), x(t-1), \ldots, x(t-k), S) \rightarrow y(t) \tag{4.27}$$

The adaptive system produces $\hat{y}(t)$ as the a priori output according to the value of $[x(t), x(t\text{-}1), .., x(t\text{-}k)]$ and the state S of the adaptive system. Basically, the adaptive system builds the function approximation of f by implementing a function g:

$$g: (x(t), x(t-1), \ldots, x(t-k), S) \rightarrow \hat{y}(t) \tag{4.28}$$

If the adaptive system can build the function g which approaches the f, the output of adaptive filter i.e. output of function g will be close to the output of f. Since the function g approaches the function f, the error $e(t)$ will be minimised and hence produces a small convergence error. At each state of adaptation, the system will produce only a small error and it approaches the convergence point faster. It is clear that the function approximation of g which maps the input space to the output space is the important part of the adaptive system. If the function approximation is very good, the error produced by the system will approach to zero. Therefore, the error correction algorithm becomes less an important consideration in designing an adaptive system.

This actual mapping function f can be a non-linear function. Therefore, a non linear processor such as artificial neural network (ANN) or a fuzzy system (FS) or combination of both of them, which approximates the function f by using non-linear function g, will produce a better result in the non-linear relation of input and output of the system than a conventional structure which employs a linear combiner.

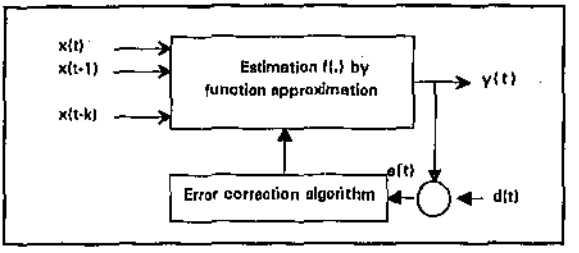

Figure 4.12 Mechanism of adaptive system

The black-box approach which is a model-free system is more appropriate in the condition when the relevant mathematical model cannot be built, especially for system identification task. Furthermore, in the condition when the details of the dynamic states of the system cannot be obtained, and only the inputs and outputs of system are available, using a black-box approach, such as a fuzzy system, is more appropriate for function approximation. It is encouraging to implement the Fuzzy-CPN model in order to perform the function approximation by learning the input-output data of the system. However, it has to be ensured that the Fuzzy-CPN will be able to perform the continuous function approximation.

4.3.1 Fuzzy neuron as fuzzy rule

Basically, a FS performs the mapping from the input space to the output space. In a conventional FS the mapping is controlled by the fuzzy rules and the fuzzy inference process. The input space and the output space are divided into r_j fuzzy regions (j = 1, 2, ... , n) which forms the fuzzy hyperbox, if every fuzzy box of the fuzzy rule base has a rule then there are $N = r_1 \times r_2 \times ... \times r_n$ rules in the fuzzy rule base. The number of fuzzy rules can be a huge number if the r_j's and n are large. However, not all rules are used to determine the final output. There are only a small fraction of these rules

that are really used in the decision of final output. The fuzzy rules which are used in the determination of the final output are called the active rules and they are applied to the particular input vector. Hence an active rule is defined as follows (Wang, 1992a):

DEFINITION 4.4

The i[th] fuzzy rule in the fuzzy rule base is *active* for $x \in Q$:

$$\text{if } m_j^i(x_j) \neq 0 \text{ for all } j = 1, 2, ..., n$$

The mechanism to activate a rule from the all the rules is implemented using a fuzzy neuron in the Fuzzy-CPN (FCPN) model to control the mapping process from the input space to the output space. The fuzzy neuron implementation has to be able to provide a basic framework to satisfy the function approximation task.

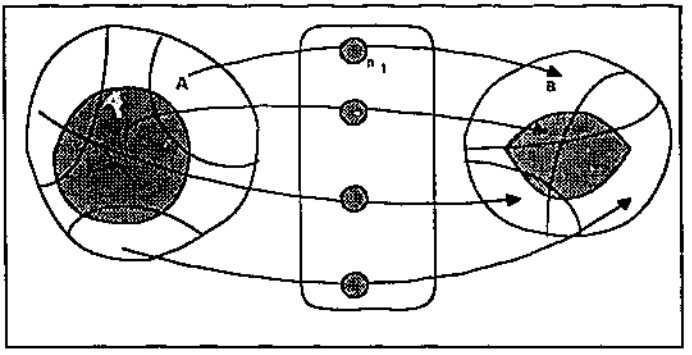

Figure 4.13 Fuzzy neuron as fuzzy rule

In Figure 4.13 , each mapping is defined as the mapping from a fuzzy patch in the input space to another fuzzy patch in the output space. A fuzzy neuron is represented as a circle which connects the input space to the output space by an arc. A fuzzy neuron controls a mapping from a fuzzy patch in the input space to the corresponding fuzzy patch in the output space. Each neuron will be sensitive to a

particular receptive field and produces a response if the input signal is in the area of the receptive field of that fuzzy neuron. The sensitive area of a fuzzy neuron is obtained during the training process as explained in the previous section. The fuzzy neuron will become active or excited i.e. the output is not equal to zero, when the input vector falls into the area which the neuron is sensitive to. Therefore the neuron performs the partition of the input space, because each neuron is only sensitive to a particular receptive field. Furthermore, the fuzzy neuron performs a mapping process which can be assigned arbitrarily. This assigning process is achieved in the learning process according to the input and output pairs as the training data by matching the flows of input vector and desired vector of the training pattern at the middle layer.

The Fuzzy-CPN performs a non-linear transformation, and it maps fuzzy sets to fuzzy sets. It behaves as the associative memory of the input space and the output space. Given an input vector in the input space the fuzzy system will produce a corresponding vector in the output space. Generally, the Fuzzy-CPN system consists of a collection of different fuzzy neuron associations. All fuzzy neurons collectively attempt to cover all area of the input space in order to satisfy the mapping from input space to the output space.

Since a fuzzy neuron performs the functions of a fuzzy rule that can be stated as:

IF *an input is in area A* THEN *output will be in area B.*

As shown in Figure 4.13, when an input vector falls into region A, neuron n_1 will become an active fuzzy neuron because it is sensitive to the area A in the input space. Hence, it yields the output value that is more than 0 and triggers the mapping process

to the area B as the mapping destination. This mechanism is similar to the active rule in a conventional FS. The output of the neuron represents the degree of fit value of input vector with respect to the rule in the fuzzy term.

Since there are some areas which overlap, there may be more than one fuzzy neuron which fire. It is different to a competitive learning neural network, which only uses one neuron as the active neuron, or a backpropagation neural network which fires all neurons with different activation levels. Since there are more than one neuron which are active but not all of the neuron actives when a particular input is being applied, the system becomes more robust and has lower complexity in comparison to the competitive neural networks and backpropagation neural networks.

Actually, the fuzzy inference process can be enriched by connecting more than one fuzzy-neuron in a special relationship. Several new operations of fuzzy neuron can be applied, such as AND, OR which have been used in the Adaptive Neural Fuzzy Inference System (ANFIS) (Jang, 1992). However, this inference strategy increases the complexity of the learning mechanism and in this ANFIS model, the number of rules has to be pre-defined.

4.3.2 Function approximation by Fuzzy-CPN model

As stated by Wang (Wang, 1992), in order to satisfy the Stone-Weierstrass theorem which proves that a Fuzzy System is a universal function approximator, some assumptions have to be satisfied. This Fuzzy-CPN model can satisfy those assumptions and it will be explained as follows.

ASSUMPTION 1

The fuzzy regions for the input and output spaces can be arbitrarily defined.

This assumption can be achieved because the receptive field of the fuzzy neurons and the centroids of the defuzzifier neuron can be assigned freely to cover the input space and output space with arbitrary size and shape.

ASSUMPTION 2

The membership functions μ_j^i can be any continuous functions from $[a_i, b_i]$ to $[0,1]$ for j=1,2, ... , n (i.e., for inputs) and from $(-\infty, \infty)$ to $[0,1]$ for j = 0, (i.e. for output), however μ_j^i must satisfy the following constraint :

$$\mu_j^i(x_j) \neq 0 \quad \text{for } x_j \in RG_j^i,\ i = 1, 2, ..., K, j = 0, 1, ..., n, \text{ with } x_0 = y$$

This constrains means that the membership value of antecedent for a rule cannot be zero if the actual input value of this antecedent falls into the required region of the rule. It can be easily achieved because whenever the input vector falls into the receptive field of the fuzzy neuron the output is always greater than 0.

ASSUMPTION 3

Any rule can be assigned to any box of the fuzzy rule base.

Since the centroid value of each defuzzifier neuron can be freely assigned and each fuzzy neuron at the middle layer can be freely connected to the defuzzifier neuron this assumption is satisfied easily.

Therefore, after satisfying those assumptions, the same steps to prove the function approximation theorem can be established in a similar way as proposed by Wang (Wang, 1992a). This existence theorem shows that there exists a way of

defining fuzzy regions, a way of choosing membership functions, and a way of assigning fuzzy rules to the boxes of the fuzzy rule base, such that the resulting mapping approximates an arbitrary non-linear continuous mapping with any accuracy. This existence theorem is similar to Kolmogorov theorem in ANN model (Hecht-Nielsen, 1987).

In this chapter, a new ANN model has been developed. This model is called Fuzzy-CPN using the Counterpropagation paradigm by incorporating fuzzy set approach at the middle layer and by implementing the fuzzy leader clustering technique. This model can be trained fast and has self-growing structure. During the learning phase, this model generates the fuzzy neurons and arranges them by adaptation in order to perform the mapping from the input space to the output space. The fuzzy neurons work as fuzzification and inference process in conventional FS. In the recall phase, by employing the defuzzifier neuron at the output layer, the final output is obtained in the parallel processing manner. This ANN architecture is used as an basic building block of the adaptive filters to perform function approximation. In the Chapter 5, this ANN model is used as an adaptive filter in the signal estimation class and be applied to solve the time series prediction. In the Chapter 6, this ANN model is used for another class of adaptive filter that is the signal correction and is applied to a noise elimination system.

Chapter 5

Non Linear Function Approximation

Function approximation is one of the ANN capabilities. Function approximation and time series prediction are important in an adaptive system as stated in Chapter 4. It has many applications such as chemical plants, cardiac pacemakers, vehicle control, sonar, manufacturing, music recording, stock market and power grids controls (Jones et al., 1989). An application of the function approximation using ANN is the control of a backing truck (Nguyen and Widrow, 1989, Kosko 1991). In this application the ANN can learn how to back a truck to a loading dock. In this chapter the Fuzzy-CPN model is used to solve the non-linear approximation problem. By employing the learning mechanism of the Fuzzy-CPN, the input-output pairs of the approximated function can be captured. In addition, by implementing the Fuzzy-CPN model as an adaptive system for the signal estimation class problem, the Mackey-Glass chaotic time series prediction is performed.

5.1 Function Approximation Problem

In Figure 5.1 in the function approximation task, given a set of examples $\{(x_1, y_1), (x_1, y_1),..., (x_1, y_1)\}$ where $y_i = f(x_i)$, the system has to approximate the function f. It can be obtained by statistical technique of the model-free approach. The generalisation is performed when the input value is not in the set of examples, i.e. at the point between the examples.

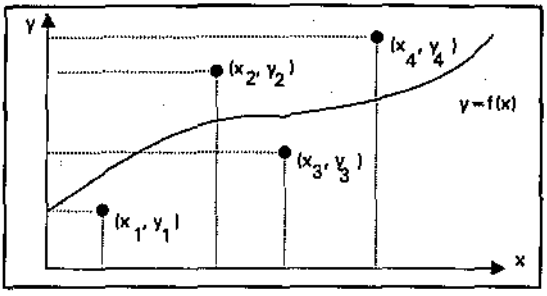

Figure 5.1 Function approximation

The simplest method is by employing the regression technique to all patterns but for some applications this technique cannot be used because it requires the basic assumption of the approximated function. Moreover, a statistical method requires an a priori knowledge about the function. However, in some applications, the input-output pairs cannot be obtained for all patterns, i.e. there are only $x(k)$ and $y(k)$ that are available at time index k. Therefore, the local information at time index k has to be used. It leads to the use of the model-free function approximation which can be performed by an artificial neural network, fuzzy system, or the combination of both of them. The neural network and the fuzzy neural network require the learning mechanism to collect the knowledge about the function f. i.e. the input-output pairs are presented into the system.

5.2 Comparison to another ANN Model

In order to compare to the conventional CPN model, the following function is used :

$$f(x) = \begin{cases} x^3 - x^2 + 0.03 & ; \text{for } x < 0.5 \\ x^3 - x^2 & ; \text{for } x \geq 0.5 \end{cases} \tag{5.1}$$

The function that is represented in Eq. 5.1 is chosen due to its non-linearity and discontinuity characteristic. To test the performance of the network, 1001 points of the function are used, they partition the input space into the same size. The input pattern is $\{x_0, x_1, \ldots, x_{1000}\}$ bounded at [0,1] and the output pattern is $\{y_0, y_1, \ldots, y_{1000}\}$. The actual function is shown in Figure. 5.2. This function is non-linear and has a discontinuity at x_{500}.

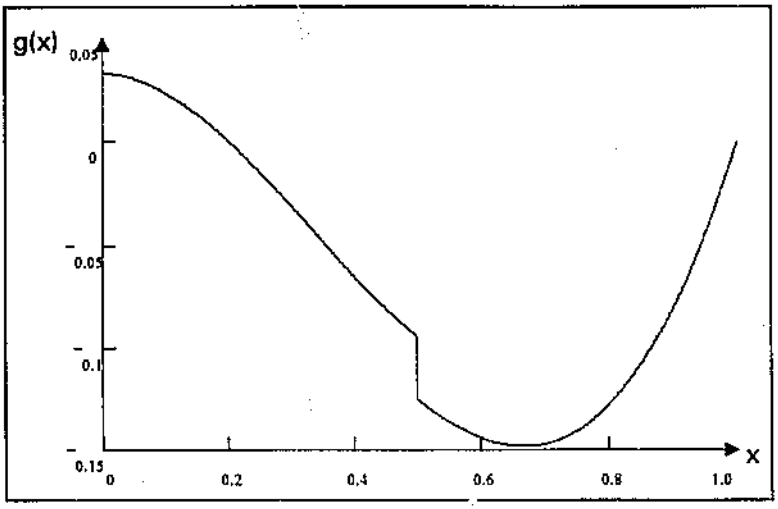

Figure 5.2 The function of Eq. 5.1.

In order to perform the comparison the mean square error (MSE) is used. It is calculated by:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \qquad (5.2)$$

where N is the number of testing data, y is the desired output value and $\hat{y}$ is the estimated output produced by the ANN model.

The training is performed to the proposed model, using 50 fuzzy neurons at the middle layer. The ρ value is equal to 0.01 and α_{limit} value is equal to 0.01. The Fuzzy-CPN (FCPN) structure is shown in Figure 5.3. It shows that the maximum number of fuzzy-neurons at the middle layer are fully exploited. However, during training, the number of fuzzy-neurons and their connections at the middle layer are varying from time to time and only during the recall phase are the number and the connections are fixed. The defuzzifier control neuron is used and denoted as the big circle in Figure 5.3.

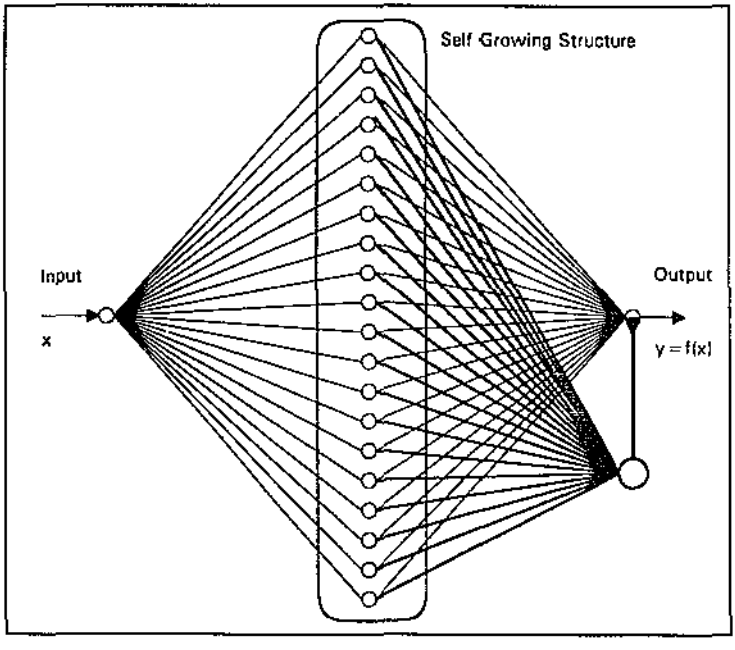

Figure 5.3 The Fuzzy CPN used

The training patterns $\{(x_0,y_0), \ldots, (x_{1000},y_{1000})\}$ are presented in the random order to the Fuzzy-CPN. After 7500 pairs of pattern have been exposed to the network, the training is stopped. After that, the input pattern is supplied to the network and the

network produces the output which approximates the given function. The result is depicted in Figure 5.4. In Figure 5.4, it is shown that the Fuzzy-CPN can approximate the function with acceptable error, and the MSE is equal to 4.906×10^{-6}. It shows that the Fuzzy-CPN can track the function $f(x)$ in most areas. The discontinuity of the function can be approximated as well by the Fuzzy-CPN without smoothing the function.

By assuming that the training patterns are represented in random order uniformly, the training patterns are assumed to be fed into the Fuzzy-CPN not more than eight times for each input-output pattern. Therefore the training time is very short. The experiment is performed using 486DX 25 MHz machine and the training time is just less than one minute.

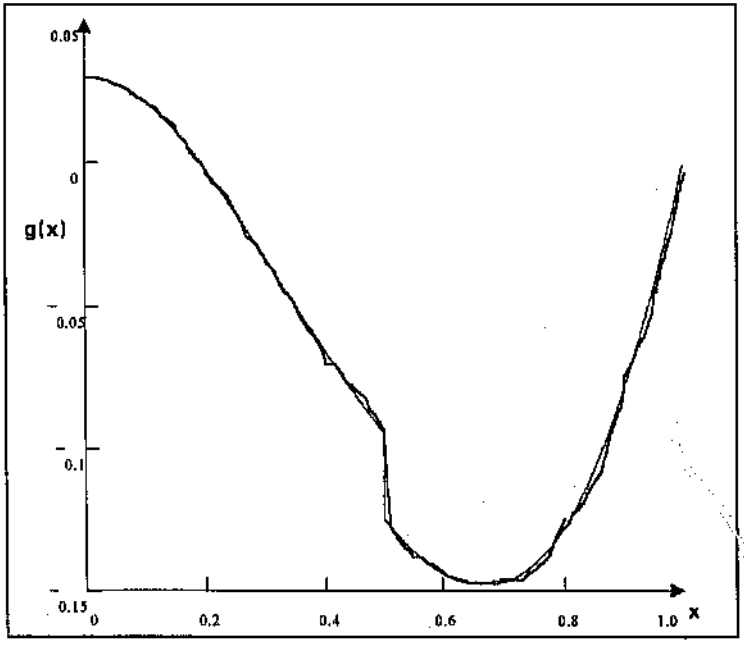

Figure 5.4 Approximation of Eq. 5.1 using Fuzzy-CPN

In order to perform the comparison in approximating a function, the two modes of Counterpropagation network are used. By using the same training pattern set the training is performed to both networks. The training patterns are presented to the network uniformly in random order. It is assumed that the representation has normal distribution. Therefore, it is able to cover the entire input space. However, for comparison it is assumed that the Counterpropagation Network (CPN) and CPN in interpolation mode (ICPN) are trained in the best way and yield the best result. For the Fuzzy-CPN (FCPN) the result is the average result after several testing cycles.

5.2.1 Comparison to Counterpropagation network (CPN)

The CPN used in this comparison has 1 neuron at the input layer, 50 neurons at the middle layer and 1 at the output layer. It is shown in Figure 5.5. The middle layer performs the competitive network using SOM paradigm.

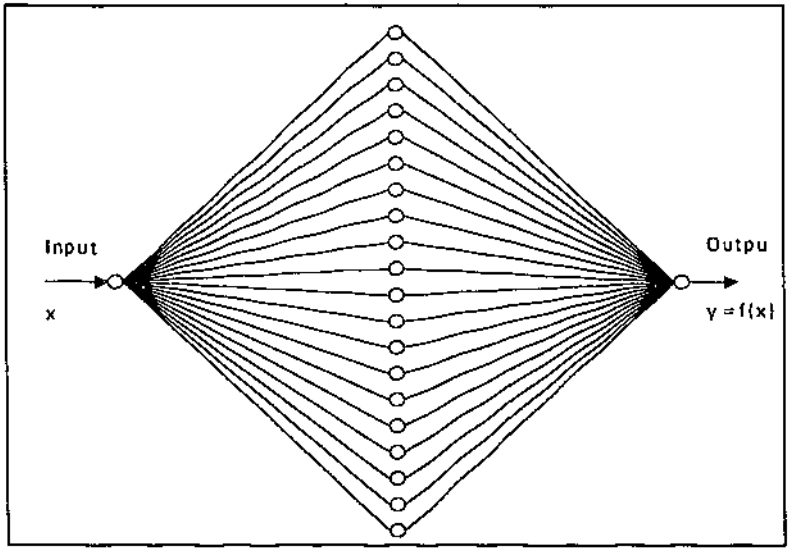

Figure 5.5 The CPN used

Training the CPN is still slow compared to the proposed model (Fuzzy-CPN). In the conventional CPN, the training has to be performed in order to build the Self Organising Map in a stable way. It requires many iterations before the SOM at the middle layer becomes stable. In the experiment, the number of iterations must be more than 1000. It takes about 10 minutes by representing all training patterns. In the Fuzzy-CPN model the number of iterations can be only 8, as long as the input space has been covered by the input pattern, and the network can approximate the function and produce acceptable result. The result of the approximation by CPN is shown in Figure 5.6.

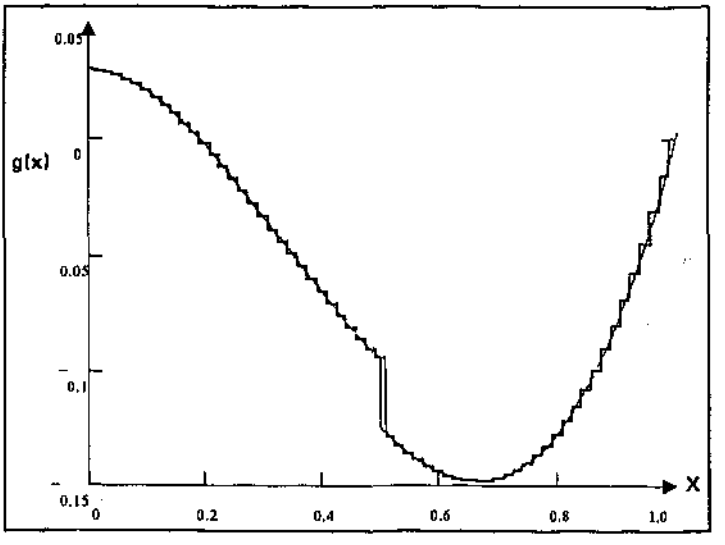

Figure 5.6 Approximation CPN to the Eq. 5.1

Assuming that this set of examples cover all the characteristics of the input-output relationship of the function, after training the CPN with this set of examples, the CPN build a look up table which has the number entries equal to the middle layer. It is equal to 50. Therefore, the output are 50 different discrete values. CPN builds the

SOM according to the input distribution. Since the input distribution is assumed to be uniform, the input space is partitioned in the same size. To the best approximation, the CPN model covers the input space with the same distribution of the input pattern. In Figure 5.6, it is shown by the same size of the partition at the x-axis. Each partition of the input space has an associated value at the output space. Therefore, the CPN produces a non-smooth function approximation. The approximation of CPN produces a jagged line. In this experiment the CPN yields MSE that is equal to 1.082×10^{-5}.

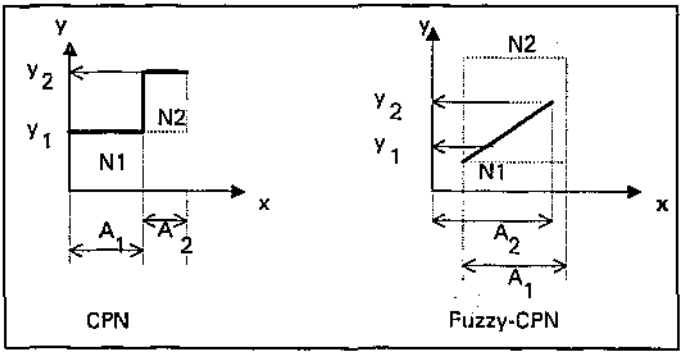

Figure 5.7 The difference approach between CPN and Fuzzy-CPN

Figure 5.7 shows the difference between the conventional CPN approach and Fuzzy-CPN (FCPN). The areas N_1 and N_2 represent the output field and receptive field of each middle neuron in the input-output space. In this example, it is assumed that there are only two neurons in the middle layer that are used. Using conventional CPN, the function produced by the conventional CPN is a discrete function because the CPN builds the look-up table only with 2 entries, where $\{(A_1 \rightarrow y_1), (A_2 \rightarrow y_2)\}$ A_1 and A_2 represents the areas in the input space. In activating the middle layer (N_1 and N_2), the competitive paradigm is implemented and the neuron which will be

triggered is only one neuron, i.e., N_1 or N_2. Both of them cannot be activated at the same time. Therefore, if there are only two neurons at the middle layer, the CPN produces only two different values of the output, i.e. y_1 or y_2.

The proposed model builds the set of fuzzy rules $\{(A_1 \rightarrow y_1), (A_2 \rightarrow y_2)\}$. In the recalling phase, the competitive paradigm is not implemented any more. It is only implemented in the learning phase to build and tune the membership functions and fuzzy rules. During recalling process, each rule which is implemented by the fuzzy neuron at the middle layer is activated in different degree and the neuron at the output layer blends them to produce a smooth line from y_1 to y_2. This approach yields an interpolation mechanism between the training patterns. It is shown that by implementing the Fuzzy-CPN, a smoother function approximation can be achieved by using less middle neurons. Hence the generalisation can be achieved without sacrificing the computational resource.

5.2.2 Comparison to the Interpolative CPN (ICPN)

A Fuzzy-CPN is similar to the CPN in the interpolative mode (ICPN) in the sense that more than one neuron in the middle layer which is active and contributes to the final ouput. In the interpolative mode, the output value of CPN is :

$$y = \frac{m_1 y_1 + m_2 y_2 + m_3 y_3}{m_1 + m_2 + m_3} \tag{5.3}$$

where y is the final output value, and y_1, y_2 and y_3 are the output of the active neurons. Three neurons which produce three-highest activation values. m_1, m_2 and m_3 are the fractional values which are bounded [0,1]. These values usually are calculated by using some heuristic approach (Hecht-Nielsen, 1987).

However, the Fuzzy-CPN is different from the interpolative mode of conventional CPN in performing interpolation between points in the output space. In the interpolative mode, in performing the interpolation between entries in the look-up table the CPN, uses linear interpolation between them. When the CPN receives an input vector and activates the middle layer, there are more than one neuron in the middle layer which win the competition between them and the winning neurons are weighted with a fraction number. The sum of all weighted numbers is equal to one. Therefore, there are the first, the second, and the third winners. According to Hecht-Nielsen (1988c), in order to perform the interpolative mode, a priori knowledge about the problem is required to define the fraction numbers i.e. m_1, m_2 and m_3. Furthermore, in the interpolative mode only three or a pre-defined number of middle neurons will be active. In the Fuzzy-CPN, the number of active neurons at the middle layer is not pre-defined and depends on the training patterns.

The partition of the input space in defining the mapping function output is performed without overlap regions for the CPN. In Fuzzy-CPN the partition of input space is performed by the overlapped areas. In Figure 5.9, A_1 and A_2 in CPN do not overlap but A_1 and A_2 in the Fuzzy-CPN do. The smoothness of the function is determined by the number of overlapped areas between the rules.

To compare the result of the interpolative mode of CPN in the function approximation, the same training pattern is input to the same CPN and the final result in recall phase is calculated by using Equation 5.3. The values of m_1, m_2 and m_3 are calculated using this Equation 5.4 :

$$m_n = \left(\sqrt{\sum (x_i - c_i^n)^2} \right)^{-1} \qquad (5.4)$$

where c^n_i is the n-th winner neuron. Therefore, it has to be determined from three templates which have the nearest distance to the input vector **x**. After that, the m for each template is calculated using Eq. 5.4. The final result of the ICPN follows Eq. 5.3. The result of the ICPN to approximate the function in Eq. 5.1 is shown in Figure 5.8:

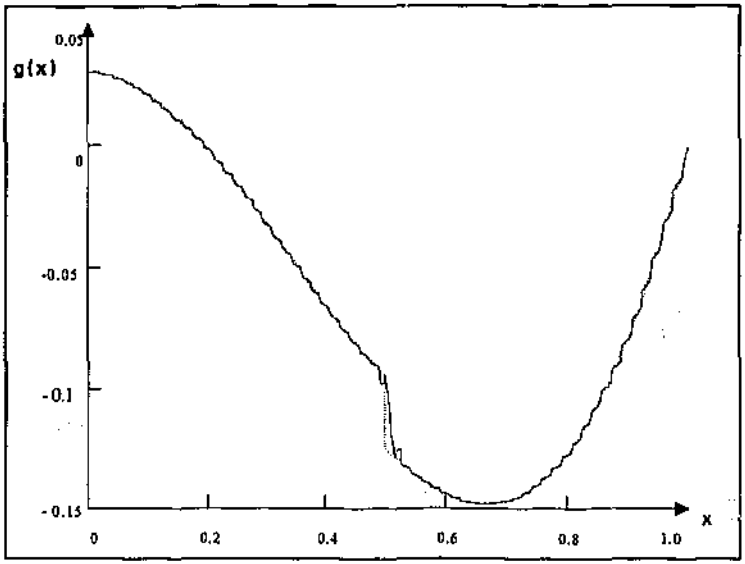

Figure 5.8 Approximation by the ICPN to the Eq. 5.1

Figure 5.8 shows that ICPN approximates between the two partitions of input space with the same approximating function. The difference between each section is the starting point and the end point. Therefore, at the area where the function has a discontinuity, the ICPN tends to approximate the function with a smooth line. It is due to the mechanism of the ICPN that always want to generalise the third closest template patterns. The Fuzzy-CPN model can arrange the fuzzy-neuron in such way

to produce the discontinuity at that area, especially when the presentation order of examples is correct. The MSE of ICPN is equal to 6.061×10^{-6}.

In addition to defining the way to calculate the fraction factor *m*, it requires extra time and a priori knowledge about the input-output relation. In Fuzzy-CPN, the learning parameter can be adjusted easily. The number of the neuron in the middle layer is set to the maximum, and the ρ is set to the small value. In the training process the Fuzzy-CPN determines by itself the numbers of neuron in the middle layer. When the result is bad and the number of neurons used in the middle layer is less than maximum, the ρ value should be increased. When the number of neurons in the middle layer is equal to the maximum, and the result is still not acceptable, the ρ value should be increased. Since the training can be executed very quickly, this adjustment process time is still acceptable.

The following figure will show the difference of the function approximation between an ICPN and the Fuzzy-CPN (FCPN). In Figure 5.9, the dots represent the training patterns. The function *f* which will be approximated is a discrete function.

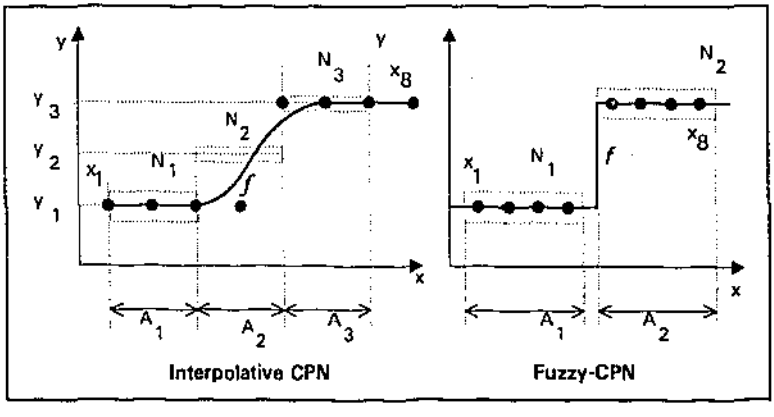

Figure 5.9. The difference approach between ICPN and Fuzzy-CPN approach.

The ICPN tends to generalise the function and produce a smooth function especially in the three nearest points. The Fuzzy-CPN produces the discrete function. In this example, the Fuzzy-CPN use only two middle neurons, where N_1 and N_2 have receptive fields which do not overlap. The output is discrete because there are only two output values, y_1 and y_2.

In Figure 5.9, it is clear that the interpolative CPN always tends to generalise all training pattern in order to produce low MSE between the three training patterns that are close to each other. It produces a function f which is a smooth line in order to approximate the training patterns $\{(x_1, y_1), \ldots, (x_8, y_8)\}$. The ICPN always approximates using the same line model shown to A_1 to A_3 for all input values. In Fuzzy-CPN, the system partitions the input space and produces the associative output value of each fuzzy neuron. In Figure 5.9, only two fuzzy neurons are used, N_1 and N_2. The fuzzy neuron N_1 produces the output as y_1 and the fuzzy neuron N_2 produces the output y_2. This mechanism provides the capability for Fuzzy-CPN to produce a non-smooth function. In addition, the Fuzzy-CPN shows a better localisation compared to the ICPN.

Using different number of neurons at the middle layer, a comparison between the CPN, ICPN and Fuzzy-CPN is performed. After the simulation is performed the ideal results of all methods are compared and shown in Figure. 5.10. It shows that the Fuzzy-CPN has the lowest MSE. To produce the same MSE, the Fuzzy-CPN requires less neurons at the middle layer.

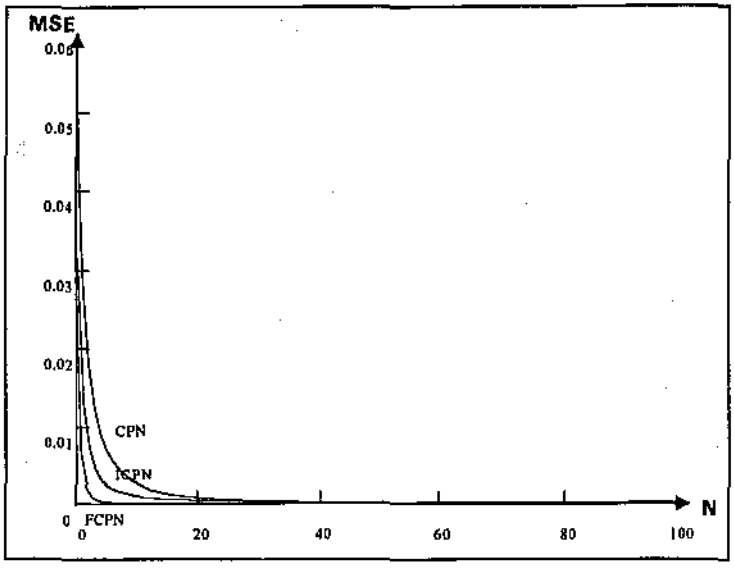

Figure 5.10 Comparison of CPN, ICPN and Fuzzy-CPN (FCPN)

It is shown that the Fuzzy-CPN can produce a discontinuous and non-linear function as stated in Eq. 5.9 by performing training with a very small number of iterations. It is suitable to the applications whenever the learning has to be done in real time. Moreover, this mechanism is suitable for on-line adaptation learning because the system does not require many iterations of learning. Especially, in time series prediction problem, after performing a prediction the real value of the next time series can be used to refine the network for predicting the next value in the time series.

5.3 The Effect of Learning Parameter to the Performance

In order to study the sensitivity of learning parameter in Fuzzy-CPN model for function approximation task, the following simple function is used :

$$y = 4x(1-x) \tag{5.5}$$

This function was used in Jones et al. (Jones et al., 1989) to test the capability of ANN model in approximating a non-linear continuous function.

The learning parameters should be easily set to suit the application. The difficulty of learning parameter setting is one of the considerations in applying a learning paradigm. However, if an ANN model requires less parameters, it is easier to tune. An ANN model does not have formalisation of tuning the learning parameters. Therefore to train a ANN model is time consuming, especially for a learning paradigm which requires many iterations for training, such as backpropagation network.

In the Fuzzy-CPN, the first learning parameter that can influence the performance is ρ, or the default value of the farthest deviation when the neuron generation occurs. The second parameter is the α_{limit}, which is the parameter that determines the neuron annihilation of a fuzzy neuron in the learning process. However, since the resource is limited, the number of fuzzy neurons at the middle layer determines the performance of the overall network itself.

First of all, the relationship parameter ρ and the number of fuzzy neurons generated during the training process N, are studied. The training patterns are the set of pair $\{(x_0, y_0), \ldots, (x_{100}, y_{100})\}$, where $x = [0,1]$ with the increment value equal to 0.01. They are given to the network which has a non-limited number of neurons at the middle layer. The training set is presented in random order and iterated for 7500 times. For the first experiment, α_{limit} is equal to 0.0. Therefore, there will be no neuron annihilation during the learning phase.

ρ is changed from 0.001 up to 0.1 with increment value equal to 0.001. After the training is performed, the test patterns are given to the network. The test

patterns are a set $\{x_0, x_1, \ldots, x_{1000}\}$ where $x = [0,1]$ with the increment value equal to 0.001. Since some test patterns have not been supplied in the training phase, this testing will show the generalisation of the Fuzzy-CPN as well. After obtaining the result produced by the Fuzzy-CPN using several ρ values the result of this experiment is shown in Figure 5.11. and Figure 5.12.

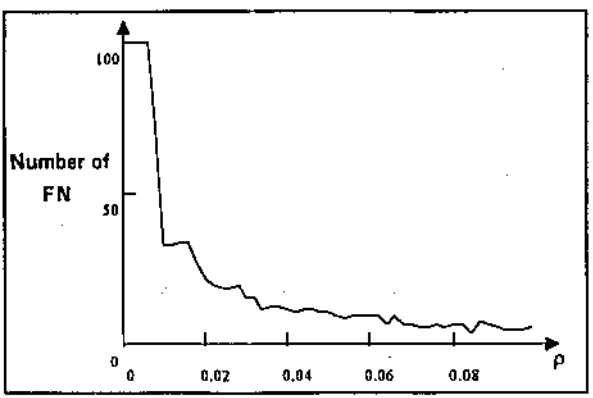

Figure 5.11 Number of neurons generated versus ρ

As shown in Figure 5.11, if the ρ value is large, there are more fuzzy neurons generated by the Fuzzy-CPN at the middle layer. It yields small partitions of the input space. However, it does not always produce better result for function approximation. In the case where the receptive field of each fuzzy neuron does not overlap the Fuzzy-CPN produces the same result as CPN. Therefore, if the ρ value is too small it yields a Fuzzy-CPN that produces the same result as CPN. It shows that the ρ value about 0.01-0.02 produce the optimum result because the number of fuzzy neuron is reduced without increasing the MSE. The relationship between the MSE of Fuzzy-CPN versus ρ is shown in Figure 5.12.

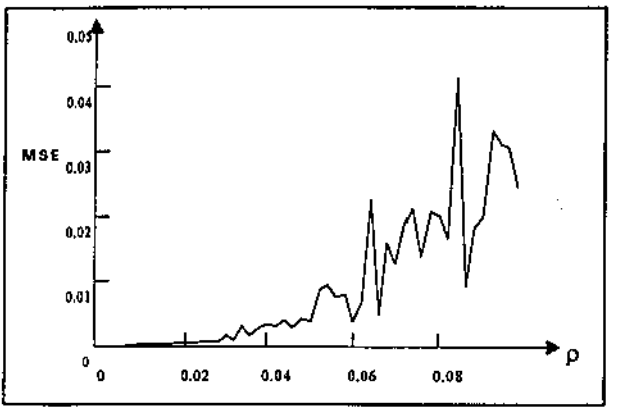

Figure 5.12. MSE versus ρ

It shows that if the ρ value is in the range 0.01-0.05 the MSE does not vary so much. After that the resulting MSE increases. Therefore for a rule of thumb the ρ value can be determined about 0.01-0.05 for 1-dimension input space.

Although using an unlimited number of fuzzy neurons at the middle layer and the small value of ρ value, an acceptable result which is equal to the CPN model still can be obtained, it produces a discontinuous function for the final result, because there are no overlapped areas in the receptive fields of the fuzzy neurons. The overlapped areas of the fuzzy neurons determine the smoothness of the function produced by the Fuzzy-CPN. Thus, to produce a continuous function it is reasonable to keep the number of fuzzy neurons at the middle layer limited so as to enable the Fuzzy-CPN to create fuzzy neurons that have overlapped receptive fields.

In order to study how to limit the fuzzy neurons at the middle layer, the following experiment is performed, using ten different ρ values. The ρ values are 0.005, 0.01, 0.015, ... , 0.05. The available memory for creating fuzzy neurons at the middle layer is limited by 10%, 20% ..., 90% of the fuzzy neurons that are

generated for the associated ρ value. This number is obtained by the first experiment. The result of this experiment is shown in Figure 5.13.

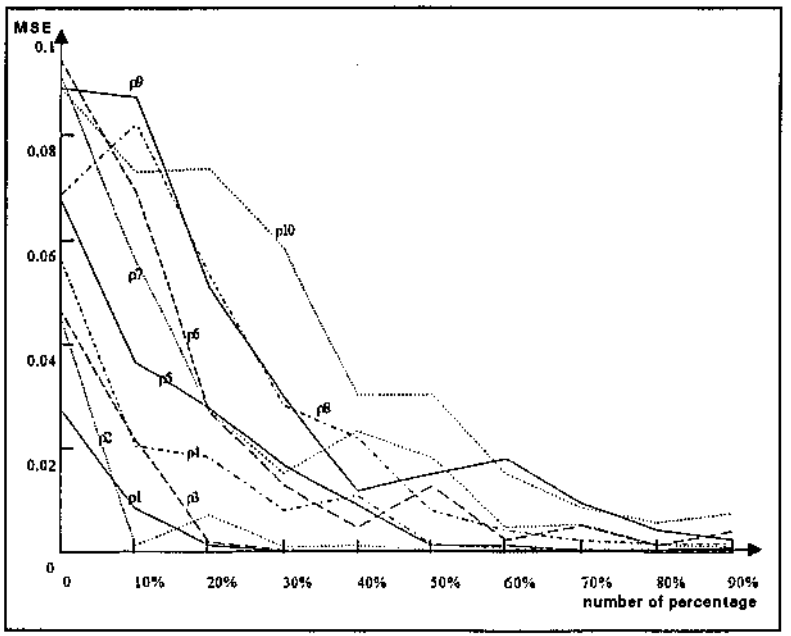

Figure 5.13 MSE versus number of neuron percentage

As shown in Figure 5.13, the MSE increases when the percentage is less than 45% and more than 75%. However, the MSE increment when the number of percentage is below 10%, is higher than the MSE increment when the number of percentage is above 75%, because if the number of neurons is limited to less than 45% the Fuzzy-CPN tend to produce a very smooth line between each samples and tends to produce a straight line or it tends to generalise or under-fit. If the number of neurons is greater than 75% the Fuzzy-CPN only tends to produce a discontinuous function to approximate the function similar to CPN. In other words, it performs too much

localisation or over-fits the function being approximated with the error bounded to the error of CPN.

The other learning parameters is α_{limit}. This value is bounded in [0,1]. This parameter determines the time when the Fuzzy-CPN annihilates a fuzzy neuron during the training phase. If this parameter is small, there will be less annihilated fuzzy neurons but if this parameter is high, the Fuzzy-CPN tries to annihilate more frequently the fuzzy neurons that are seldom used. In another word, the Fuzzy-CPN begins to forget the patterns learnt before. This value determines the behaviour in this self-growing structure network. To study the influence of this value to the overall performance of the network, the following experiment is performed. For this experiment, ten different values of ρ are used, with the same values as in the previous experiment. According to the previous result, the fuzzy neuron are limited to the 60% of the maximum number for each ρ value. Since this value tends to produce the lowest MSE. By varying the α_{limit} for 0.05, 0.01, ..., 1.0, the training is performed. The result of this experiment is shown in Figure 5.14.

Figure 5.14. shows that the variation of α_{limit} which does not effect so much the performance of the network. After all, if the α_{limit} value is in the range 0.57-0.65 it decreases the MSE of the Fuzzy-CPN. The α_{limit} value does not produce significant improvement but it is useful in applications where temporal training data are required, for example time series prediction. The order of training sequence itself determines the influences of the α_{limit} to the network performance. By keeping this α_{limit} minimal, the Fuzzy-CPN works without performing any annihilation process.

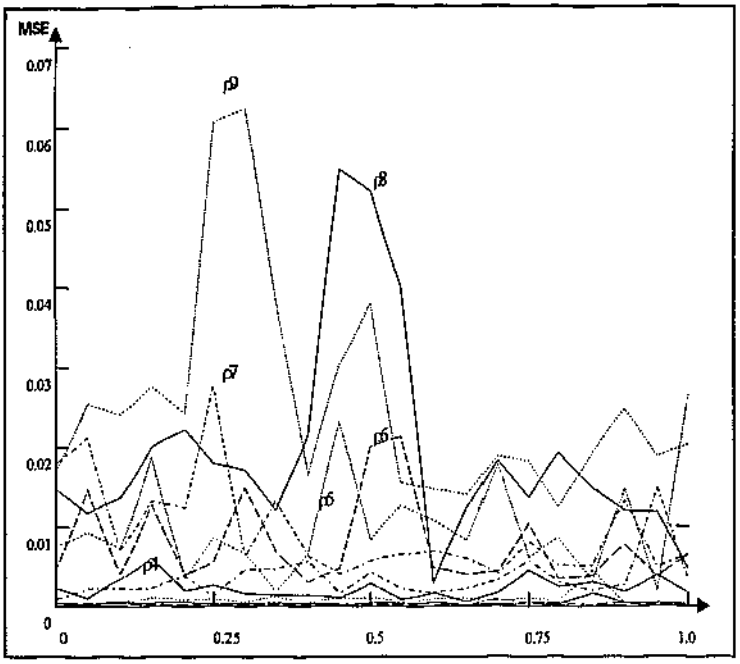

Figure 5.14 MSE versus α_{limit}

It has been shown that the learning parameter of this network is easy to adjust and it does not affect the overall performance much. Thus this Fuzzy-CPN is easy to build and train. To adjust the learning parameter, the following steps are used:

- The number of middle neurons is set to be unlimited.
- Set the ρ in the range 0.01-0.02 and choose ρ value that produces the lowest MSE with the smallest number of fuzzy neurons.
- Reduce the number of available resource for middle layer to the 50%-60% of the maximum fuzzy neuron number generated for the chosen ρ.
- To obtain further improvement, the α_{limit} can be set to be equal to 0.6.

5.4. Time Series Prediction

To apply the capability of the Fuzzy-CPN to perform a non-linear function approximation, a simulation of the Fuzzy-CPN model is used to solve the time series prediction problem. Using some past data, the prediction of the future data is performed. The prediction is achieved by performing the function approximation from the past data to produce the future data. Suppose f is the function to estimate a future value, in time index k:

$$f: [x(k), x(k-1), ...x(k-p)] \rightarrow x(k+q) \tag{5.6}$$

where p determines the number of the past data that are used to estimate the future data and q determines the future value that will be predicted. Therefore, the function f is a $(p+1)$-dimension surface, in the $(p+2)$-dimension space. To estimate the future value $x(k+q)$, the values $\{x(k), ..., x(k-p)\}$ can be applied to the function f. Now the task becomes estimating the function f by using a set of examples.

In this section, the Mackey-Glass chaotic time series will be used to test the performance of the network. This problem is chosen because it is a benchmark problem that has been cited quite often in literature, (Roger et al, 1993, Wang and Mendel, 1992, Jang, 1992, Jones et al., 1989).

Chaotic time series is a deterministic and non-linear series. Mackey-Glass chaotic time series is generated from the following delay equation :

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{5.7}$$

where the τ value determines the chaotic behaviour of the function. In this simulation, τ equal to 30 is used. This value is similar to the model that has been used by Wang and Mendel (Wang and Mendel, 1992). This time series is shown in

Figure 5.11. This time series is not a random time series but it is a deterministic model that has chaotic behaviour. Therefore, it seems like a random series.

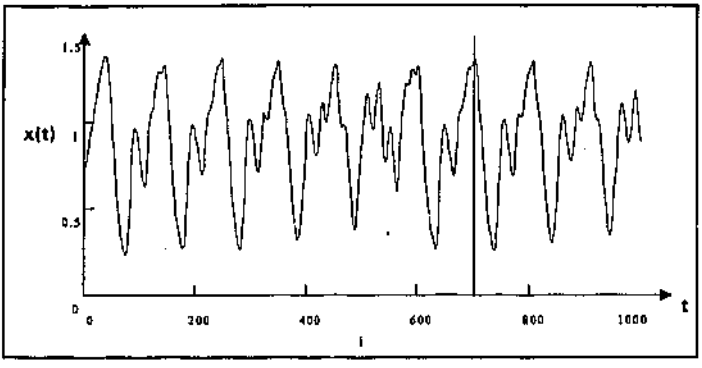

Figure 5.15 Mackey-Glass chaotic time series

In this application, to predict the future value of the time series, in Eq 5.6. p is equal to 8 and q is equal to 1. This means that the future value is predicted by using 9 past values. The network is given the input [$x(k)$, $x(k-1)$, ... , $x(k-8)$], and it has to predict the next value $x(k+1)$. It means that the network perform a function approximation of function f, where :

$$f: R^9 \rightarrow R \tag{5.8}$$

$$f: [x(k), x(k-1), \ldots, x(k-8)] \rightarrow x(k+1) \tag{5.9}$$

Thus the Fuzzy-CPN used in this simulation has 9 neurons at the input layer and 1 neuron at the output layer. The maximum number of fuzzy neurons at middle layer is equal to 80 neurons. In the simulation, the training is performed without repeating the training patterns. The ρ value is equal to 0.0125 and α_{limit} value is equal to 0.7.

The simulation is performed in two different learning strategies. The first strategy is to train the network by using all training patterns. Then, the Fuzzy-CPN is

used to estimate the rest of the future data. For the second strategy, after the Fuzzy-CPN estimates a future data, the actual future data is used to retrain the Fuzzy-CPN for further estimation. However, the retraining process is performed without repeating all previous training patterns, and only the last actual data are used.

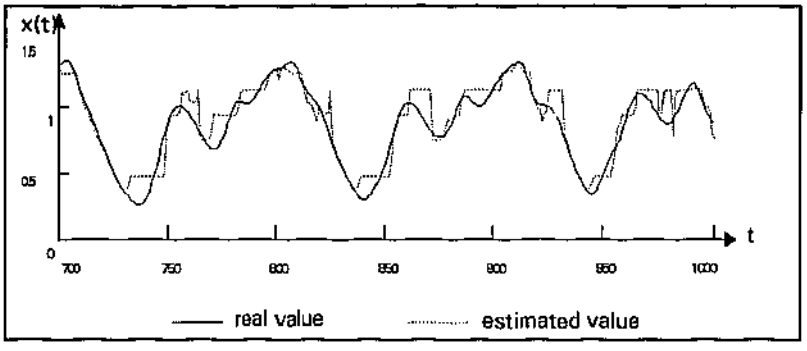

Figure 5.16 Simulation using x_{500} to x_{700} as training data

The first simulation is performed using the first strategy or without on-line adaptation. The first 200 data from x_{500} to x_{700} are used as the training patterns. The prediction is performed for the rest of data x_{701} up to x_{1000}. The result and the comparison with theoretical calculation are shown in Fig 5.16.

The prediction is not too close enough and the MSE is equal to 0.010715. This result is obtained because the network has not been trained with enough samples and only 60 fuzzy neurons are created in the middle layer. It shows that the Fuzzy-CPN has not been able to cover the input space, shown by some parts of the output that form a flat line even though the input changes. The overlapped areas of the receptive fields of fuzzy neurons determine the smoothness of the final output. For instance, they are shown in the point x_{740} - x_{750}, x_{770} - x_{780}, and the other parts. In some points, the estimation has not be able to follow the dynamic of the system. The

changes of the estimation are left behind the actual output. For example, This is shown in the area x_{750} - x_{760}, x_{965}- x_{985}.

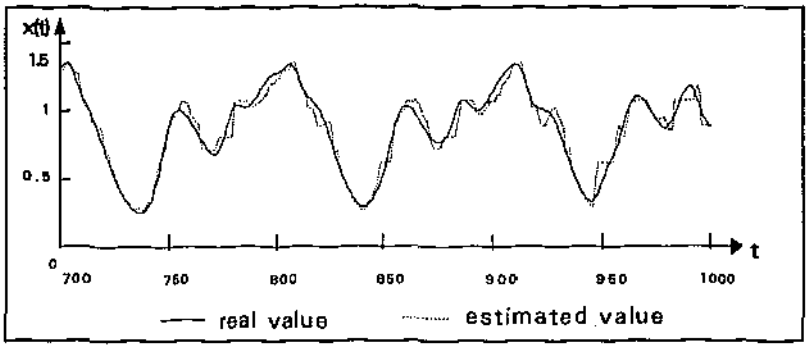

Figure 5.17 Simulation using x_0- x_{700} as training data

The second simulation is performed by using the first 700 data as the training patterns. The result is shown in Fig. 5.17. By comparing it with the theoretical result, it can be shown that the system performs prediction better than the previous experiment. The MSE is equal to 0.0035749. After being trained with 700 data, the Fuzzy-CPN has been able to perform the estimation and to capture the dynamic of the system. This result is quite similar to the result that were obtained by Wang and Mendel using backpropagation and Fuzzy System (Wang and Mendel , 1992), or Jang using ANFIS (Jang, 1992). However, in both works they used iterative training and in this work only one pass training is used. It proves the capability of the Fuzzy-CPN to perform very fast training.

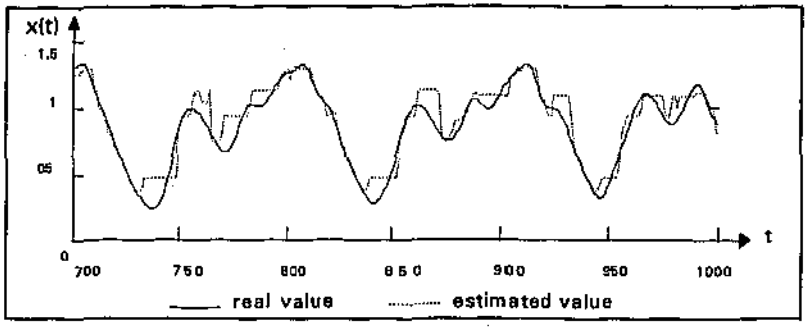

Figure 5.18 Simulation using on-line adaptation x_{500} - x_{700} as training pattern

In Figure 5.18 and Figure 5.19 the second strategy, the adaptive on-line training is used. When the on-line training is used for the case that the network has already been trained using the past 700 data, there is no great difference, because the network has converged, and the mean square error is 0.00357490. It shows that the adaptive training does not significantly affect the performance of the network. When this mechanism is applied to the network which has only been trained from 200 data, there is a large improvement. The mean square error is reduced from 0.010715 to 0.009049. This mechanism is suitable for learning in the real-time application because the Fuzzy-CPN requires only a small number of iterations to produce an acceptable result.

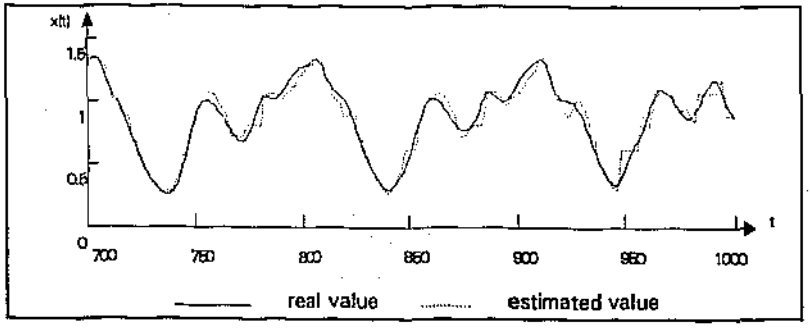

Figure 5.19 Simulation using on-line adaptation x_0 - x_{700} as training pattern

The experiments show that the capability of the proposed model to perform function approximation. This function approximation capability is suitable for the adaptive filter task by performing the signal estimation class. The result is promising and shows that the Fuzzy-CPN has good generalisation and localisation when applied for the prediction task. The learning parameters do not affect the performance so much. Therefore, the setting of learning parameters is easy and not sensitive to the network performance. The prediction of Mackey-Glass chaotic time series has been demonstrated by applying the proposed network. In addition, it has been shown that the on-line update capability of this network provides the error reduction in the normal operation. This on-line adaptation is hard to implement by using a learning mechanism which requires an iterative learning scheme. Therefore, this Fuzzy-CPN model is suitable for real time applications, e.g. in the problem where the network has to be trained before it will be used, in order to capture the change of the environment when the system is used. This proposed method will be further expanded to tackle noise elimination problem which will be described in Chapter 6.

Chapter 6

Background Noise Elimination

The elimination of background noise in applications where an uncorrupted input signal is required is not a trivial task, especially when the noise is non-deterministic and non-stationary. This chapter addresses the application of the Fuzzy-CPN model as an adaptive filter that performs a signal enhancement in a noise elimination system which makes use of the time-frequency representation of the input signal and a spectral subtraction for filtering process. Software simulation results with some typical signals corrupted by Gaussian noise are also presented.

6.1 Noise Elimination

In many areas of science and engineering in which input signals are obtained using signal acquisition equipment, such as electromagnetic and acoustic surveillance (Anderson et al., 1990, Casellman et al., 1991), seismic signal processing (Magotra et al., 1991), speech processing (Kobatake et al., 1990), biomedical signal processing (Uncini et al., 1990), there is always a problem in detecting the presence of non-stationary random signals or noise in the background. In addition, the duration of this noise may be short compared to the observation intervals for the input signals

Enhancement of a noisy signal means raising the SNR of the signal. It is a mapping process from the noisy input signal space to the noise-reduced output signal space. This mapping can be linear or non-linear, and time-variant or time-invariant depending on signal characteristics (Trompf, 1992). Various methods have been

proposed to tackle this problem including fixed filtering (Boll, 1979) and adaptive filtering techniques using conventional filters. In the case of fixed filtering techniques, the unavailability of an ideal signal model and the use of predefined parameters of the filter makes this problem difficult to solve (Classen et al., 1983). The use of adaptive methods have improved the performance of background noise elimination to a large extent (Vary, 1983). With these methods, the parameters of the filter change to adapt to the non-deterministic characteristics of the background noise. However, provision has to be made for the filters to readapt to the change in the environment. Sometimes it may lead to the unsatisfactory performance of the overall system (Connel et al., 1990).

The conventional adaptive algorithm such as LMS algorithm with 32-tap FIR structure has been used for an Adaptive Periodic Noise Cancellation (APNC) for the control of acoustic howling for the hand-free telephone situation. However, for other applications which require the elimination of non-linearity and non-Gaussian noise do not produce satisfactory results (Wright and Foley, 1979).

6.1.1 Noise elimination model

The objective of speech enhancement is reduction of noise level, increase in intelligibility or reduction of the auditory fatigue (Cheng, 1991). Several techniques such as adaptive noise cancelling, spectral subtraction, adaptive comb filtering may be used. Figure 6.1 shows a general model for noise elimination where the input sequence $s(k)$ represents the desired input signal and $u(k)$ any external noise signal. Inside the model, $\omega(k)$ represents internal broadband measurement noise. The problem becomes one of accurately modelling the corrupted channel $\alpha(.)$ and the

internal noise by means of the filter β(.). In general, α(.) represents a non-linear medium. Therefore, to achieve the best modelling of α(.), β(.) must be a non-linear function.

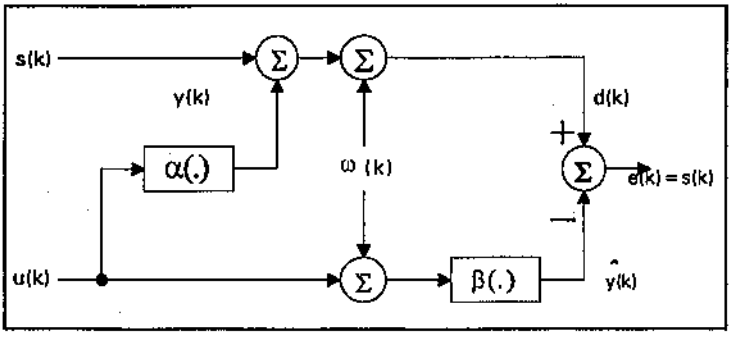

Figure 6.1 General model of noise elimination

In the example of speech processing, the noise elimination process is to detect and eliminate the noise from the non-stationary speech source. In Figure 6.1, the signal received by the system is $d(k)$ which is the noisy version of the signal $s(k)$. The noise term is produced by $u(k)$ which is any external noise signal and $\omega(k)$ represents the deterministic and stationary internal broadband measurement noise. Therefore, the signal $d(k)$ is:

$$d(k) = s(k) + \alpha(u(k)) + \omega(k) \tag{6.1}$$

The sequence $u(k)$ itself has two components :

$$u(k) = c(k) + m(k) \tag{6.2}$$

where $u(k)$ is the background noise that can be considered as consisting of a non-deterministic and stationary noise $c(k)$ and non-deterministic and non-stationary noise isolated noise $m(k)$.

$$\hat{y}(k) = \beta(u(k) + \omega(k)) \tag{6.3}$$

In order to eliminate noise, the model tries to model the channel α by using the function β. The final output of the noise elimination model is the difference between the received signal and the noise estimated by the function β.

$$e(k) = d(k) - \hat{y}(k) \tag{6.4}$$

For the best estimation, the original signal can be recovered :

$$e(k) = s(k) \tag{6.5}$$

Such assumption should be taken to differentiate between the effect of the degradation process and the good input signal. These differences can be used to regenerate the good signal from the corrupted one. However, the problem will arise when the assumption fails. This assumption limits the algorithm that can be applied. The perfect restoration of the mixing of two signals which are random is impossible, since the mixing represents a loss of information (Betts and Reid, 1993). The algorithm will generate only the most likely good signal.

The scope of the implementation of this model has been restricted to the following assumptions :

- Source selection : recording of the noise and signal is assumed to be performed with a single microphone without microphone modification. Therefore, two microphones approach (Kiperztok, 1993) should be ignored. Moreover, it is assumed that the listener cannot control the microphone. Therefore, the transducer does not have any modification to reduce the noise.
- Noise characterisation : only additive noise uncorrelated with the clean signal will be considered. This approach will not try to model the background noise statistic.

- Non speech distortion and cochannel distortion will not be considered as well (Boll, 1983).

These assumptions are similar to those which have been stated by Boll (1983) for the Noise Suppression Problem.

6.1.2 Background noise detection

The speech signal is a very highly non-linear signal (Hambaba et al., 1990). Therefore, it is hard to extract it from the noise using analytical methods. The noise and the signal are assumed to be generated by two different sources: the signal source and the noise source. For an analytical method, in order to extract the signal from the noise, a knowledge about the relation) signals is required. Therefore, to determine the relationship between the noise and the signal, the relationship between the sources plays the important role. Two considerations should be taken into account in determining the relationship between two signals in terms of their primary sources. Firstly, it has to be known whether the signal is related by single sources, dependent sources or independent sources. Secondly, if the two signals are related by a common source, it should be known if there is a time sequential relationship between them. The signal produced by a primary source can be highly independent of one another. However, it depends on the nature of the sources themselves (Brandt, 1991).

To determine the linear relationship between two signals, a frequency domain method such as coherence analysis can be employed. It is obtained by dividing the estimation of the cross-power spectra of the two signals by the product of their auto-spectra as a function of frequency. More complex analytical methods involve the

use of periodicity, pitch frequency, Linear Predictive Coding (LPC) has been proposed by Kobatake et al. (Kobatake et al., 1990), for the restoration speech contaminated by non-stationary noise. However, in many signal analysis problems, curvilinear regression cannot be applied effectively due to the fact that the order of the polynomial fit may be too high and the statistical characteristics of the signal changes in time, which means that the order of the polynomials should be calculated whenever the signal changes.

In an effort to detect signal without many prior assumption about the signal and the medium, a blind identification method is developed (Tong et al., 1991). Blind identification has been motivated by practical problems that involve multiple source signals and multiple sensors which share a common objective, i. e. separating and estimating the source signals without knowing the characteristics of the transmission channel. The characteristics of the medium is unknown because the relative position of the multiple sources are not known a priori. There are many conventional algorithms for blind identification task such as Extended Fourth Order Blind Identification (EFOBI) that assumes the source signals are independent. However this algorithm cannot handle Gaussian sources. The Algorithm for Multiple Unknown Signal Extraction (AMUSE) is developed in order to tackle the Gaussian noise and for non-white signals only. However, some basic assumptions should be taken into account to apply these algorithms to the real-world problems. Different algorithms require different environments and assumptions to be able to be applied successfully. For example, the MUSIC algorithm (Multiple Signal Classification Method) has better result than Maximum Likelihood Method (MLM) or Maximum Entropy Method (MEM) in detecting a signal. However, it requires the background noise to

be uncorrelated (Jeffries and Farrier, 1987). The MEM is suitable for modelling a smooth spectra but will not perform well on line spectra. Other methods, such as MUSIC, models the signal as a number of line spectra and will not be able to model the ocean noise.

It leads to the idea for making use of the adaptive filter for performing blind deconvolution (Blesser and Kates, 1978). It is expected that the system will be able to adjust itself with respect to the change of environment. Some models such as Adaptive Noise Cancelling (ANC) using adaptive algorithm by Widrow (Furukawa and Kubota, 1990) only removes the low frequency sinusoidal noise during transmission. Therefore, for those methods it is always assumed that a good signal model is always available. In the contrary, this proposed model does not assume the good signal model to be available but assumes that the noise samples are available.

6.1.3 Time-frequency representation as pre-processing

Detection and classification of signal components in time or frequency domain can be performed by firstly sliding the signal through an observation window. Normally, to extract the signal feature from either the time or frequency domain, representation of the signal is not sufficient for a complete analysis of the signal. Therefore, a combined time and frequency representation is used in this proposed model. In this proposed model, the modified speech spectrogram is used. A speech spectrogram contains rich acoustic knowledge about speech. Therefore, it is a good tool in analysing speech (Flanagan, 1972).

Some work has been done in combining ANN with the time-frequency representation in signal pre-processing. The backpropagation (BP) and Discrete

Fourier Transformation (DFT) that performs channel filter bank is used to discriminate voice/non-voice for integrated packet switching (Shimokoshi and Hashitsume, 1989) and speech segmentation for a continuous speech recognizer (Finster, 1992).

Instead of implementing a single huge ANN model to perform the complete task, the proposed model uses time-frequency representation of the signal to pre-process the input signal before it is processed by the ANN. The implementation of a single ANN is impractical because training the huge network is time consuming and difficult (Botros et al., 1992). The time-representation of the signal reduces the work of ANN to perform the complete task. This mechanism is quite similar to the mechanism of the human ear to extract the interesting sound from the environment noise (Eisenberg et al., 1989). In order to process the acoustical images with neural network of the brain, both kinds of information the time and frequency information are required.

The power spectra of the speech provides a good initial representation of speech. It could be discovered with minimal a priori assumptions about what are meaningful transformation or representation data. The result is better than when the network is presented with unanalysed digitised waveform (Hambaba et al., 1990). The speech spectrogram as the time-frequency representation method which combine the time representation and the frequency representation of the signal can provide data pre-processing for a neural network model. Although, the representation of the spectrogram appears more complicated in 2-dimension, it makes the network easier to process in order to detect and classify the signal. It is shown by Palakal and Zoran (Palakal and Zoran, 1989) that the multi layer network models will be able to learn

the invariant features from the speech spectrogram. The RBF-Net can be trained to extract known discriminatory features in speech patterns from speech spectrogram (Niranjan and Fallside, 1990). Based on these evidences, the spectrogram approach is used as the time-frequency representation in the proposed model for noise elimination using Fuzzy-CPN.

There are several methods to generate the time-frequency distribution of a signal (Cohen, 1992). The method adopted in the proposed model is the spectrogram approach. Given a signal $s(t)$, corrupted by noise, its time-frequency distribution is expressed as:

$$\rho(t,f) = \int\int\int e^{j2\pi v(u-t)} g(v,t) s^*(u-\tfrac{1}{2}\tau) s(u+\tfrac{1}{2}\tau) e^{-j2\pi f\tau} dv du d\tau \qquad (6.6)$$

where g is the kernel function and s^* is the complex conjugate of s.

For the spectrogram representation, the kernel function is:

$$g(v,\tau) = \int h^*(u-\tfrac{1}{2}\tau) e^{-j2\pi vu} h(u+\tfrac{1}{2}\tau) du \qquad (6.7)$$

where h is a window function and h^* is the complex conjugate of h.

By substituting Eq. 6.6. to Eq. 6.7., the conventional spectrogram approach for time-frequency representation of a signal is written:

$$\rho(t,f) = \left| \int e^{-j2\pi ft} s(\tau) h(\tau - t) d\tau \right|^2 \qquad (6.8)$$

In this work, the proposed spectrogram method is performed by performing further transformation form the result of Eq. 6.8. This modification is done to ensure the time-shift invariance and the time-origin invariance properties of the time-frequency representation of the signal. The proposed spectrogram method is defined in Eq 6.9:

$$\zeta(\theta,f) = \int |\rho(t,f)|^2 e^{j2\pi\theta t} dt \qquad (6.9)$$

To give the example of this transformation, suppose $s(t)$ is the signal which will be used to illustrate the time-frequency transformation:

$$s(t) = \begin{cases} A_1 \sin(\omega_1 t) & \text{for } t < p \\ A_1 \sin(\omega_1 t) + A_2 \sin(\omega_2 t) & \text{for } p \le t < l \\ A_2 \sin(\omega_2 t) & \text{for } l \le t \end{cases} \qquad (6.10)$$

The signal $s_1(t)$ and $s_2(t)$ contain a stationary sinusoidal signal with frequency equal to ω_1 and a sinusoidal chirp with frequency equal to ω_2. These signals are shown in Figure 6.2. and Figure 6.3, respectively. The difference in the signals in the two figures is the starting point of the chirp with the frequency ω_2.

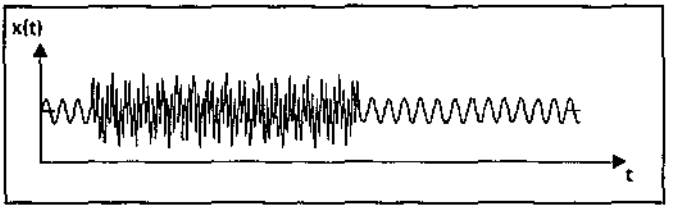

Figure 6.2 Signal $s_1(t)$ in Eq.6.10 with $p = 56$, $l = 300$

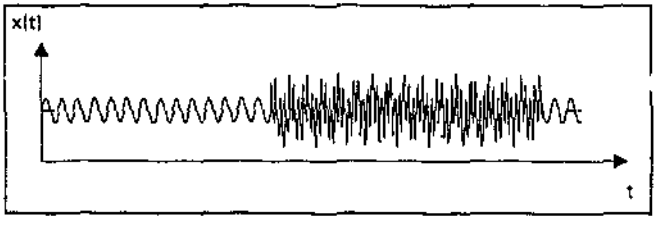

Figure 6.3 Signal $s_2(t)$ in Eq.6.10 with $p = 256$, $l = 300$

As shown in Figure 6.2, the signal $s_1(t)$ has a chirp that starts at the point t_{56} and in Figure 6.3, it shows that $s_2(t)$ has a chirp starting at the point t_{256}. Both signals are transformed using conventional spectrogram using Eq 6.8 and displayed in Figure 6.4. and Figure 6.5. for a 3-dimension representation, the x-axis represents the

frequency component of signal, f, the y-axis represents the time, t and the z-axis the magnitude of the signal component in a particular frequency, G, which is the result of Eq. 6.8. To make the representation clearer the corresponding contour diagram are shown in Figure 6.6. and Figure 6.7, respectively.

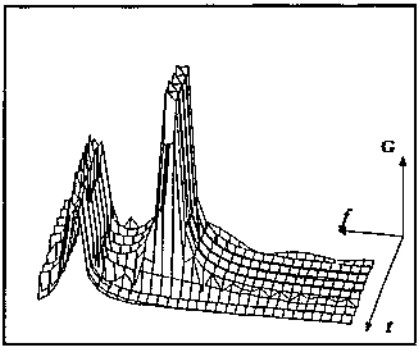

Figure 6.4 Conventional Spectrogram of $s_1(t)$

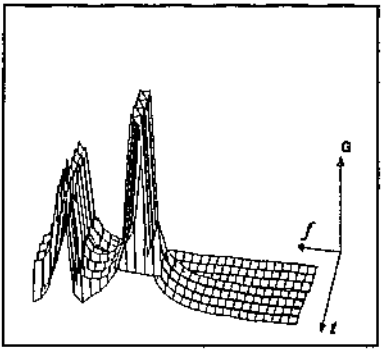

Figure 6.5 Conventional Spectrogram of $s_2(t)$

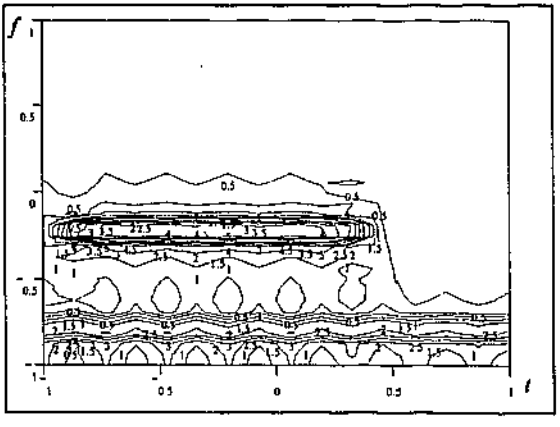

Figure 6.6 Contour diagram of Figure 6.4

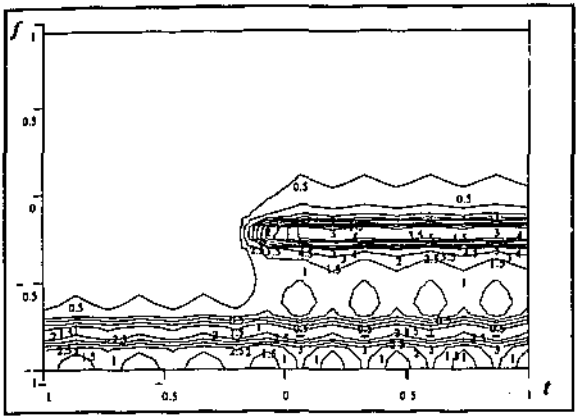

Figure 6.7 Contour diagram of Figure 6.5

The conventional spectrogram representations of the two signals are different, even though the original signals are not different. The cause that produces the difference is only the starting point of each frame of transformation. If the both signals are transformed using proposed method in Eq. 6.9 the result are Figure 6.8 for $s_1(t)$ and Figure 6.9 for $s_2(t)$ The x-axis represents the frequency component of signal, f, y-axis represents the changing of a particular frequency component, θ and z-axis represents the result of transformation , G, defined by Eq. 6.9.

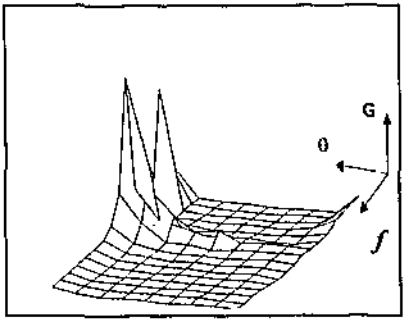

Figure 6.8 Proposed spectrogram of $s_1(t)$

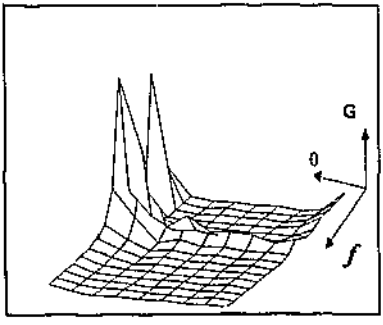

Figure 6.9 Proposed spectrogram of $s_2(t)$

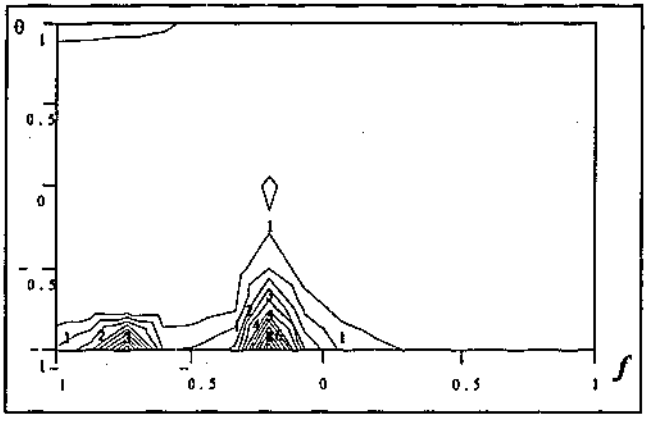

Figure 6.10 Contour diagram of Figure 6.8

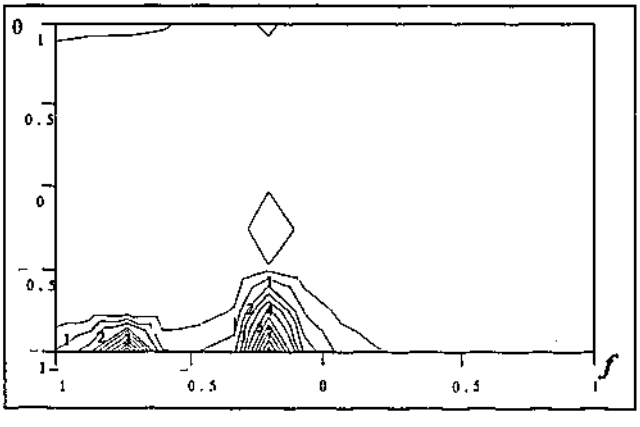

Figure 6.11 Contour diagram of Figure 6.9

By using the proposed model, the spectrogram representation does not produce significantly different representations of them. It reduces the complexity in detecting the signal, especially for non-stationary signals.

This formulation has been used to decompose a signal into different components (Cohen and Lee, 1992) using one Short Time Fourier Transform (STFT) for each non-overlapped block of signal. The optimal choice of window for the spectrogram must depend on the signal involved and its characteristics at the time

of observation (Cohen and Lee, 1990). More complex feature extraction techniques such as Karhunen-Loeve which has been extended by Fukunaga and Koonts will provide better result for a feedforward network to classify the input signals (Shaudy and Leen, 1992), especially for the spectrum of signal as the raw data. Wavelet transform as another option for time-frequency representations and has been applied for pitch detection of speech signal (Kadambe and Bartels, 1992). The proposed spectrogram approach is chosen due to its simplicity and low computational complexity, because it is based on the Fourier Transform. Therefore, it can be easily implemented using FFT algorithm. In the Appendix C the time-frequency representation of real noises that are recorded from the sample CD are shown using the proposed method.

6.1.4 Spectral processing for the noise elimination

The final filtering process of the corrupted signal itself is performed in the frequency domain. It means that the system performs the filtering process by manipulating the spectrum of the original signal. The spectral subtraction is used to remove the noise power spectral component. For the spectral subtraction, a direct estimation of the short term spectral-magnitude is required.

Speech is assumed to be a random process and the added noise is uncorrelated. A speech signal contains a noise can be written as:

$$x(k) = s(k) + n(k) \tag{6.11}$$

The noise is assumed to be stationary in short-term. Since it is assumed that x and s are stationary process for limited time and n is uncorrelated.

$$P_x(\omega) = P_s(\omega) + P_n(\omega) \tag{6.12}$$

where they are the power density spectrum (PDS) of $x(k)$, $s(k)$, and $n(k)$, respectively. Since they are only stationary within a time limited frame, the short-time power spectra are related by:

$$P_x^i(\omega) = P_s^i(\omega) + P_n^i(\omega) \tag{6.13}$$

where the superscript i is a frame index. This short time model of the speech can approximate the non-stationary signal (Cheng, 1991). However, in the conventional techniques, the noise is estimated with the second order during the silent-times (Deller, 1992) or from reference channel (two microphone system).

If P_s and an estimate of $P_n(.)$ are available, it is possible to estimate the power density of the uncorrupted signal. The estimated noise power spectrum signal is subtracted from the transformed noisy input signal.:

$$\hat{P}_s(\omega) = P_x(\omega) - \hat{P}_n(\omega) \tag{6.14}$$

Using DTFT, in the window size of m, :

$$\hat{\Gamma}_s(\omega; m) = \Gamma_x(\omega; m) - \hat{\Gamma}_n(\omega; m) \tag{6.15}$$

$$\left|\hat{S}_s(\omega; m)\right|^2 = \left|S_x(\omega; m)\right|^2 - \left|\hat{S}_n(\omega; m)\right|^2 \tag{6.16}$$

The filtering process of the noisy speech in general by using this approach can be shown in the following Figure 6.11. Details of this techniques can be found in Boll (1979), Deller et al, (1992). This method starts by performing a DFT to transform the signal corrupted by noise into the frequency domain representation. After performing the noise spectra estimation, the spectra of signal corrupted by noise is modified by subtracting the estimated noise power spectra. Finally, the IDFT is performed to yield the time domain representation of the noise-free signal.

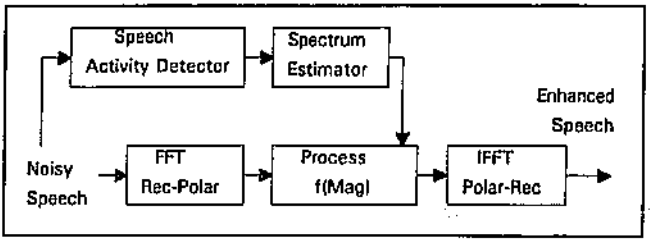

Figure 6.12 Speech enhancement process using STFT (Boll, 1987)

This method was first developed by Weiss et al. (Weiss et al., 1974) and used for suppressing tones, clicks and broadband additive noise. This spectral subtraction method has improved the intelligibility score as shown by Boll (1979). It has been used to restore old music records (Blesser, 1978).

In this noise elimination model, the Fuzzy-CPN performs the estimation of the noise spectra in the signal. This estimation is performed by exploiting the associative memory capability of the Fuzzy-CPN. However, an additional process is still required in order to produce the final output of the filter. It can be done using spectral subtraction method (Boll, 1979), or adaptive comb filter or by using a structure such as graphic equaliser.

Since the noise is not stationary for a long time period. A method for estimating the noise spectrum is required. The choice for this model is the average periodogram. Periodogram is one of the cost-effective techniques for estimating the speech signal (Godwin et al., 1988).

Define a truncated signal :

$$x_n^N = \begin{cases} x_n & \text{for n = 0, 1, 2, . . N - 1} \\ 0 & \textit{otherwise} \end{cases} \tag{6.17}$$

The estimation of the power spectrum of this truncated signal is:

$$S_N(\omega) = \frac{1}{N} X_N(\omega) X_N^*(\omega) = \frac{1}{N} |X_N|^2 \tag{6.18}$$

where the transforms of both the original and truncated signal are :

$$X_N(\omega) = \sum_{-\omega}^{\omega} x_n^N \, e^{-jn\omega T_s} = \sum_{n=0}^{N-1} x_n \, e^{-jn\omega T_s} \tag{6.19}$$

to yield a consistent estimator of S $\{x(n),\ n = 0,\ 1,\ \ldots\ N\text{-}1\}$. X_N is divided into K segments, each of length M = N/K.

$$X^{(i)}(n) = x(iM + n) \qquad \begin{matrix} n = 0,\ 1,\ \ldots\ ,\ M - 1 \\ i = 0,\ 1,\ \ldots\ ,\ K - 1 \end{matrix} \tag{6.20}$$

Periodogram of each segment of length M is given by:

$$\hat{I}_M^{(i)}(\omega) = \frac{1}{M}\left|\sum_{n=0}^{M-1} x^{(i)} \, e^{-jn\omega T}\right|^2 \tag{6.21}$$

The average of the periodogram is:

$$S_M^K(\omega) = \frac{1}{K}\left|\sum_{i=0}^{K-1} \hat{I}_M^{(i)}(\omega)\right| \tag{6.22}$$

The average of the periodogram is a consistent estimator and unbiased estimator of the power spectra. $x(n)$ is assumed to be stationary and I_M is an asymptotically unbiased estimator. Therefore, S is asymptotically unbiased as well. The correlation between $x(n)$ and $x(m)$ will be weaker as the difference between m and n become larger.

In estimating power spectra using periodograms, a finite number of the input signal sample is used by applying rectangular window to the input signal. The number of the input signal sample in that window determines the bias of the periodogram to estimate the power spectrum. The bias is inversely proportional to the number of the input signal sample. Furthermore, by using a finite number of the input signal sample, the periodogram produces poor frequency resolution

To choose the number of the input signal sample in a window that is used to estimate the PSD of the signal, there is a trade-off between the choice of K and M in Eq. 6.20. For fixed N, once K is chosen so is M = N/K. If K is larger, the reduction in the variance of $\hat{S}_M^K(\omega)$ will be larger, but it has a larger bias and poorer frequency resolution. For a smaller K, the reduction in variance will be smaller, but $\hat{S}_M^K(\omega)$ has a smaller bias and better frequency resolution.

In order to reduce the trade off between the variance and the frequency resolution, the block are partially overlapped the sequence. It reduces the bias of the periodogram in estimating the power spectrum because the correlation between $\hat{I}_M^{(i)}$ and $\hat{I}_M^{(j)}$ is stronger when they are adjacent blocks that overlap. Consequently the reduction of the variance of $\hat{S}_M^K(\omega)$ will not be large as large as in the case of non-ovelapping blocks. It produce smaller bias and better frequency resolution.

The process of calculating the periodogram is performed by exploiting the Fourier Transformation which is implemented using the Fast Fourier Transformation (FFT). The FFT algorithm has been developed for a long time and explored widely since Coley-Tukey released their paper (Coley and Tukey, 1965). Those algorithm are Coley-Tukey Algorithm, Rader-Brenner Algorithm, Good-Thomas Algorithm, Goertzel Algorithm, Winograd Algorithm., and the other modifications. The FFT algorithm can meet the lower bound on the multiplicative complexity of DFT (Duhamel, 1990). The algorithm has been improved by digit-reversal permutation (Evans, 1987), split radix (Duhamel, 1986). For computing the inverse DFT, FFT can use the forward DFT (Duhamel et al., 1988).

Cyclic convolution of real data can be performed efficiently using Fourier Transform algorithm (Duhamel and Veterli, 1987). Therefore the final filtering

process is performed by implementing the spectral subtraction and the final result is obtained after executing the Inverse FFT.

6.2 Fuzzy-CPN for Background Noise Elimination

Artificial Neural Networks (ANNs) offer an alternative technique for adaptive filtering. In the area of background noise elimination, Xue (Xue et al., 1992) has applied an ANN-based adaptive matched filter in biomedical processing. Lippman et al. (Lippman et al., 1989) have shown the capability of ANN to perform adaptive pre-processing for signal detection in non-gaussian noise. The noise reduction problem has been considered in the development of robust speech recognition (Trompf, 1992).

Unsupervised learning can learn the hidden structure of speech (Hambaba,et al., 1990). The network can develop a rich internal representation and the learning provides a systematic way to find the features in data. It has been applied in many areas such as the pitch detection of an acoustic source (Cohen et. al., 1992) (Cole et al., 1991), initial data reduction to acoustic data in the engine fault detection (Hewitt et al, 1989), the electroencephalogram classification (Tsoi et al., 1993). the fault testing on the helicopter gear box using the vibration wave of machine (Rock et al., 1993), and a spectral processing of harmonic complex tone and pitch (Tomlinson and Treurniet, 1990). The application of ANN model for signal detection and classification is due to the fact that it will be able to reduce the computational complexity.

Backpropagation networks (BP) have been used for the adaptive noise filtering by using the spectral data as the input data to remove white noise from the

input signal. Although, BPs are successfully in reducing the overall error in obtaining a pure signal from noisy spectrum, they fail to pay attention to the details of the spectrum (Weber et al., 1991). Moreover, BPs have three major problems. Firstly, the energy surface of cost function has many local minima. Secondly, it is difficult to analyse the behaviour of the hidden units. Thirdly, the BP learning algorithm is extremely slow. Conventional backpropagation networks cannot handle input data of a large dimension and not capable of extracting spatial features. Moreover, they cannot be applied effectively to extract the signal feature from the 2-dimensional FFT spectrum (Palakal and Zoran, 1992)..

Fuzzy logic has been used for Active Noise and Vibration Control (ANVC). (Kipersztok, 1993). This approach uses two microphones, one for the noise only and another for the combination of source signal and noise signal. It has been applied to some areas such as noise control in rooms. However, it still uses an adjustable-FIR structure which is controlled by a fuzzy controller.

Another method for applying the ANN model is the Orthonormal Neural Network (ONN) using the Fourier method to remove noise from the corrupted signal (Ulug, 1992). The speech enhancement auditory evidence method can be applied for stationary and non-stationary noise. The advantages of this method is that it does not need any a prior knowledge about the noise, and only a modest computation is required (Cheng et al., 1991).

6.2.1 Structure of the model

The proposed fuzzy neural model should be able to satisfy some application constraints:

- There is no clean speech signal for target patterns, which should be derived from the input patterns. The network cannot be trained using a clean speech signal;
- Learning must be performed as fast as possible with a small number of iterations;
- The neural network should be able to adapt its structure to minimise memory usage;
- Fast processing time is required for real-time operations, i.e. the network has to be implemented in parallel processing.

In the proposed approach as shown in Figure 6.12, the Short-Time Fourier Transform (STFT) is first used to get the frequency representations for overlapped blocks of signal samples. Then another STFT is performed at the same frequency to all the resultant frequency representations to obtain the change of the frequency of the signal. The result of this manipulation is to extract finer representation of the frequency feature of the signal. The final outcome of the transformation of the signal is a two-dimensional array as shown in Figure 6.9 or Figure 6.10, of a single time block of the input signal. This array is then fed into the neural network.

Basically the noise elimination performs three main steps in eliminating the noise in the corrupted signal .

- Detection of the noise;
- Estimating the noise power spectra;
- Performing spectral manipulations to produce the uncorrupted signal.

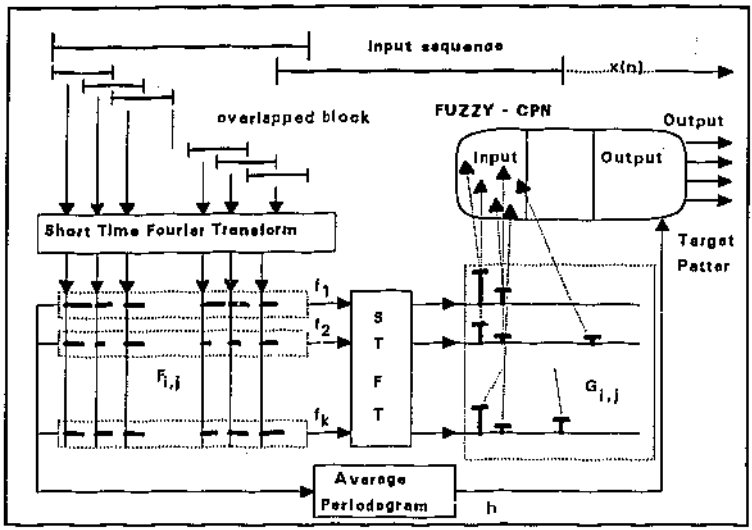

Figure 6.13 Input signal acquisition process.

These steps are rather similar to the auditory evidence method that was proposed by Cheng (Cheng 1991), which incorporates the simulation of the ear's enhancement model. This method makes use of spectra average and performs the distance measure between spectra. The calculation involves direct and inverse fourier transforms, spectra convolution and time domain convolution (Cheng, 1991). The other model which is similar to the proposed model is the signal enhancement model that has been applied for audio restoration in music recording, i.e. to remove click and scratch. It works by detecting the click and scratch, thus removes them by performing an interpolation of the signal in time series. It has been used in the Gerrard MRM-101 Music Recovery Module using signal interpolation techniques (Reid, 1989). However, the difference between this model is that the detection and the estimation of the noise spectra are performed using a trainable system Fuzzy-CPN without taking any assumption of the noise and the signal model.

The operation of the proposed system consists of two phases, the first is training phase and the second is the recall phase. In the training phase, the system picks up the background noise and trains the system to build the template of the noise which will be used to estimate the noise power spectra in the recalling phase. Each template is encoded as a fuzzy neuron in the middle layer and each of them is associated with a model of average periodogram as the power noise estimation of the noise. The average periodogram is encoded as the weights of the defuzzifier neuron. This training phase has to be done in real time with very small number of iterations because, every time the system is used, it has to be trained to model the background noise of the environment where the system is used.

In the recall phase, after being input with the corrupted signal, the Fuzzy-CPN estimates the noise power spectrum of the signal by using associative characteristic of the Fuzzy-CPN. After that, this information is used to process the corrupted signal, and is fed into the adjustable filter. The adjustable filter can be formed by using Finite Impulse Response (FIR) filter, Infinite Impulse Response (IIR) filter or by using resonator bank filter (Sztipanovitz, 1989). In this work the spectral subtraction method is used in the frequency domain. Since the frequency domain representation of the input signal is already available from the time-frequency representation, frequency domain filtering is performed. However, this structure has a problem in the block boundaries, and may lead to small discontinuities at the filter output (Shynk, 1992). To an application such as speech, this disadvantage does not significantly effect the human testing. In our model, the effect of small discontinuities in the output signal is reduced by overlapping the blocks of input samples.

Suppose $[x(k), \ldots, x(k+M)]$ is a block of the input signal samples, where $x(k)$ is the input signal at time index k, and $M+1$ is the number of input samples for one block of signal. The first transformation is executed to the frames $[x(k), \ldots, x(k+N)]$, $[x(k+L), \ldots, x(k+L+N)]$, ..., $[x(k+PL), \ldots, x(k+PL+N)]$, where N is the number of sample in a frame for the first Short Time Fourier Transformation (STFT), L is the number of overlapped samples, P is the number of frames in a block which are processed by the STFT.

Each successive block frame produces the magnitude of the STFT ($F_1, \ldots, F_M$) and for all frames in one block they are denoted as :

$$[(F_{1,1}, F_{2,1}, \ldots, F_{Q,1}), (F_{1,2}, F_{2,2}, \ldots, F_{Q,2}), \ldots, (F_{1,R}, F_{2,R}, \ldots, F_{Q,R})].$$

where for F_{ij}, the index i in the subscript represents frequency number as the result of the STFT. The index j denotes the frame number. The conventional spectrogram approach stops at this point and produces this matrix **F** as the final result.

For further step of the proposed transformation, the second STFT is performed to the F which has the same i-index. Therefore, the data are rearranged into this structure :

$$[(F_{1,1}, F_{1,2}, \ldots, F_{1,R}), (F_{2,1}, F_{2,2}, \ldots, F_{2,R}), \ldots, (F_{Q,1}, F_{Q,2}, \ldots, F_{Q,R})].$$

Each block $(F \ldots F)$ is processed using the STFT again and it yields the other matrix :

$$[(G_{1,1}, G_{2,1}, \ldots, G_{S,1}), (G_{1,2}, G_{2,2}, \ldots, G_{S,2}), \ldots, (G_{1,U}, G_{2,U}, \ldots, G_{S,U})]$$

Therefore, the matrix **G** represents the changing of signal at particular frequency or the changing of the frequency component of the signal. The matrix **G** is fed into the

Fuzzy-CPN system as the input signal. The value of first STFT ($F...F$) is used to calculate the average periodogram in order to estimate the noise power spectra of the training patterns. Let **h** be the average periodogram of the signal from the {$x(k)$ ... $x(k+M)$} and **h** is calculated using Eq.6.22. The frames of signal sequence are shown in Figure 6.14.

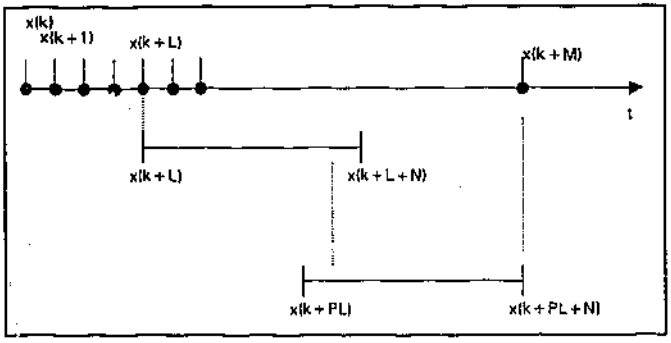

Figure 6.14 Frames of the signal

For this Fuzzy-CPN, the set of training patterns is {($\mathbf{G}_1$, $\mathbf{h}_1$), ($\mathbf{G}_2$, $\mathbf{h}_2$), ... , ($\mathbf{G}_n$, $\mathbf{h}_n$)}. This set of training patterns is supplied to the Fuzzy-CPN. The training is performed using **G** as the input signal and **h** as the desired signal.

During the recall phase, after an input signal is given [$x(k)$,..., $x(k+M)$], the same steps are performed to produce the time-frequency representation of the input signal $\mathbf{G}_{in}$. The Fuzzy-CPN produces the average periodogram $\mathbf{h}_{out}$ as the noise power estimation of the signal corrupted by noise. After that, this **h** information is used to perform the spectral subtraction. The result of the spectral subtraction is the spectrum of the noise-reduced signal.

6.2.2 Fuzzy-CPN as associative memory

As stated in the previous section, to eliminate the noise from the corrupted signal, the system performs the mapping from the signal corrupted by noise onto the noise-reduced signal space. In performing this mapping process, the system acts as an heteroassociative memory (Rumelhart, 1989). Given a time-frequency representation of the signal **G**, the system produces the average periodogram **h** of the noise associated with the received signal including noise. This average periodogram is used to perform spectral manipulation of the signal.

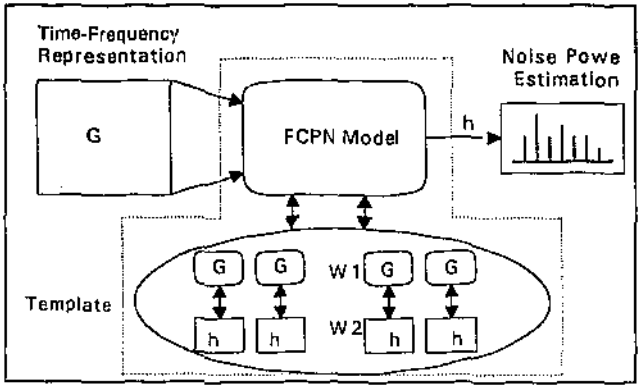

Figure 6.15 Fuzzy-CPN (FCPN) as associative memory

In Figure 6.15, basically, the Fuzzy-CPN determines the time-frequency representation **G** of the signal and can classify it into one class of templates. In the Fuzzy-CPN, those templates are encoded in the weight **W1**. Since the Fuzzy-CPN employs the fuzzy approach, in this classification process, it does not produce a template as a winner. It produces a membership value of the input signal to be classified into each template. The periodogram of each template itself is encoded in the **W2** as the centroid of each class.

In the Fuzzy-CPN, there is one to one correspondence of **G** to **h**, i.e. $\{(\mathbf{G}_0 \rightarrow \mathbf{h}_0), (\mathbf{G}_1 \rightarrow \mathbf{h}_1), \ldots, (\mathbf{G}_N \rightarrow \mathbf{h}_N)\}$. It is clear that they actually represent a set of the fuzzy rule as :

IF *input is* $\mathbf{G}_i$ THEN *output is* $\mathbf{h}_i$

However, since the system uses fuzzy set approach in determining the output which relates to its template, the output is a combination of the templates or the new model of the template. The final output itself can never exist in the set of $\{\mathbf{h}_0, \mathbf{h}_1, \ldots \mathbf{h}_i\}$ or a new form of **h**.

The Fuzzy-CPN has the ability to do novelty detection (Kohonen, 1990). It is different from the conventional CPN which uses a winner take-all structure at the middle layer. For the associative memory, the output value of conventional CPN is limited to the number of neurons at middle layer. It means that the capacity of the conventional CPN is equal to N, where N is the number of neurons in middle layer (Singh et al., 1992). The capacity, that is the number of patterns that can be stored without the model performs an interpolation in the recall phase, in this proposed model is not limited by the number of neurons in the middle layer. Thus, the capacity is greater than N. The capacity is influenced by the overlapped areas of the receptive fields of the fuzzy neurons in the middle layer. It is due to the fact that the function produced by the Fuzzy-CPN becomes smoother whenever the fuzzy neurons have the overlapped area amongst each other.

It can be viewed that the fuzzy neurons, besides performing IF-THEN rules, work as the attractors in the input space. After training is performed, the fuzzy neurons relax as a number of attractors. There is evidence that the representative

inner states do not necessarily correspond to any real input states. In other words, it does not necessary mean that an attractor represents a class of real noise. This associative capability of Fuzzy-CPN is exploited to perform the estimation of the noise power spectra of the signal corrupted by noise.

6.3. Simulations

The software simulation for the system has been performed and the results are given below. For simulation purposes, Gaussian noise are used for training the network. The size of the network used consists of 16x16 nodes for the input layer, only 20 neurons in the middle layer, and 16 neurons in the output layer. The ρ is equal to 0.1 and the α_{final} is equal to 0.25.

The measurement of the filter performance is represented by using Signal to Noise Ratio (SNR). For this experiment, the SNR is calculated using

$$SNR = 10\log\left(\frac{P_{signal}}{P_{noise}}\right) \tag{6.23}$$

where P_{signal} is the power of signal and P_{noise} is the power of noise. The simulation results are performed by using off-line simulation. The noise is obtained by generating the Gaussian noise using DSPWorks® software (Momentum Data System, 1992). The noise elimination process is performed by the software simulation written in PASCAL. After training the Fuzzy-CPN with this noise, a signal corrupted by noise is given to the system. The result of noise elimination is shown in Figure 6.17.

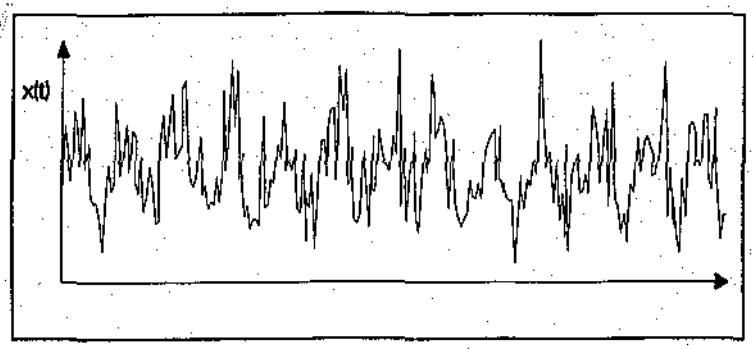

Figure 6.16 Sinusoidal input signal corrupted by Gaussian noise with SNR equal to 17.49 dB

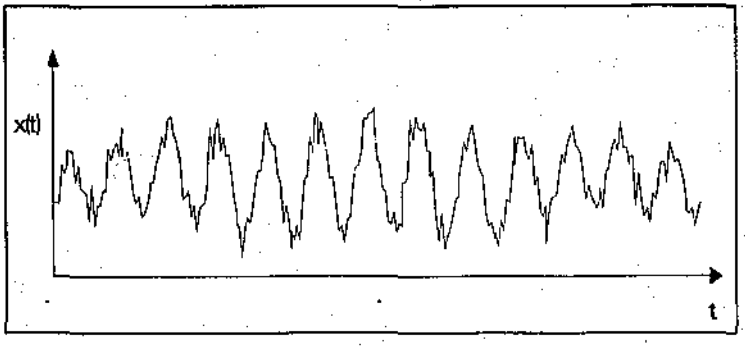

Figure 6.17 Output signal after filtering, with SNR equal to 27.07 dB.

Figure 6.16 shows the input sinusoidal signal corrupted with Gaussian noise with SNR equal to 17.49 dB and Figure 6.17 the resultant output signal from the system with an improved SNR equal to 27.07 dB. It is shown that the corrupted signal can be recovered and only a small portion of noise still exists in the signal. However, the remaining noise can be eliminated by using a low pass filter.

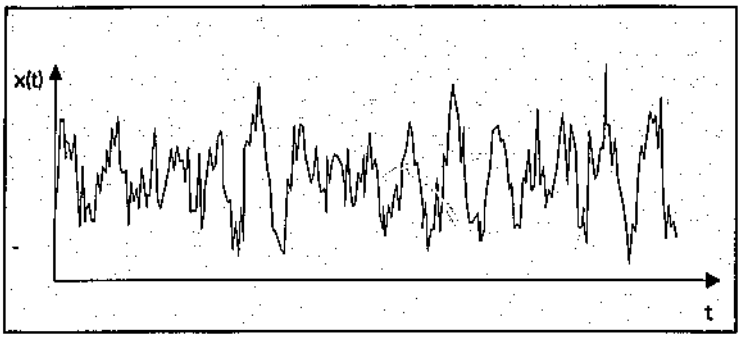

Figure 6.18 Sinusoidal input signal with amplitude modulated, with SNR equal to 20.31 dB

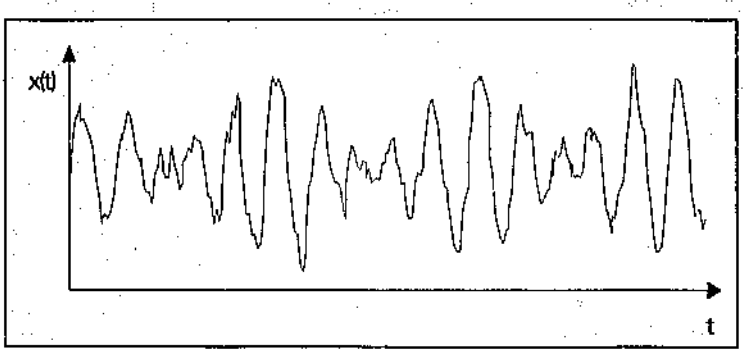

Figure 6.19 Output signal after filtering, with SNR equal to 25.52 dB.

To test the magnitude response of the system, a sinusoidal signal which has been amplitude modulated, as shown in Figure 6.18, is fed into the system. The original signal is almost recovered with SNR 25.52 dB, which is shown by Figure 6.19, with its amplitude unchanged. However, there is a small boundary problem in this simulation.

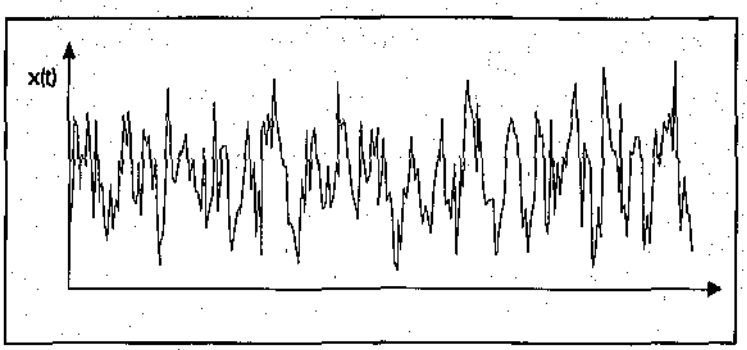

Figure 6.20 Sinusoidal input signal with frequency modulated, with SNR equal to 12.37 dB

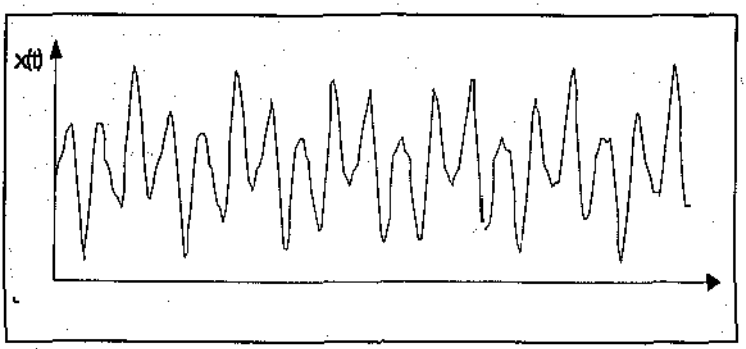

Figure 6.21 Output signal after filtering with SNR equal to 29.03 dB.

The frequency selectivity of the system is tested by a frequency modulated signal as shown in Figure 6.20. The results in Figure 6.21 shows that the original signal is recovered but with a small phase shift. The reason for the phase shift is the use of frequency domain filtering for the filter. The problem can be solved by using resonator bank filters instead of frequency domain filter, or making use of the phase information of the noise when the spectral subtraction is performed.

To test the system with the real noise, some real noises are used. For investigation purposes, a 1 KHz sinusoidal signal is used as the desired input signal. The noises are sampled from CD by using TMS320C25 DSP Card (Dalanco Spry,

1993) which is controlled by DSPWorks® software (Momentum Data System, 1992). After the system has been trained using the noise, the sinusoidal signal mixed with the noise is input to the system. The system tries to eliminate this noise and produce the desired signal. The detailed result of this experiment are shown in Appendix C. The results show a significant improvement.

These simulations show that the proposed noise elimination model can work effectively in the situation where the SNR of the corrupted signal is low. This condition makes the system more easily detect and estimate the noise power spectra in order to perform spectral subtraction. When the SNR of corrupted signal is high, the signal becomes dominant compared to the noise. It creates difficulty for the Fuzzy-CPN to estimate the noise that corrupts the signal. Furthermore, for correlated noise, this method cannot produce a good result, because the spectral subtraction work under assumption that the noise is uncorrelated with the signal.

This proposed model for noise elimination produces a promising result to be applied in the real time application using a dedicated signal processor system. The system makes use of both time and frequency representation of a signal as input to a neural network based adaptive filter. The formulation of the time frequency representation has been given and modifications of the transformation process using overlapped blocks of signal has produced the frequency representation of the signal to overcome the boundary problem. In addition, the use of a second short-time Fourier Transform on the spectrogram to extract finer details on the frequency changes of the signal results in better representation of the frequency feature of the signal.

Chapter 7
Conclusions

7.1 Summary of Results

The speech enhancement process is required because speech transmission and processing are often degraded by acoustic or electrical noises. This process is expected to improve the quality and intelligibility of the received signal. In the non-linear problem, most of conventional linear adaptive filters technique, such as Least Mean Square (LMS) or Recursive Least Square (RLS), cannot tackle the speech enhancement task with satisfactory result.

Basically, an adaptive filter performs the adaptation process by performing function approximation to produce an a priori ouput, followed by a particular error correction algorithm to reduce the error between the a priori output and the desired signal. Therefore, by implementing a non-linear function approximator, an adaptive filter which can solve a non-linear problem can be built. Furthermore, a function approximation capability of the adaptive processing should be considered as the main aspect because it dictates the error produced by adaptive system.

ANN has the capability to learn the input-output relation in the non-linear relationship and requires less assumptions to build the model. The fuzzy approach is built up from the possibility theory and may be used to deal with the ambiguity or the vagueness. Both ANN and Fuzzy System (FS) are model-free estimators that can estimate a function without knowing the mathematical model of the input-output relation in performing a mapping from input space to the output space. It is ensured by the existence theorem from Kolmogorov for ANN and the Stone-Weierstrass

theorem for the FS. The learning capability of ANN model can reduce the development time and the cost of designing the fuzzy system. A combination of both techniques yields a learnable system that can deal with the vagueness problem.

A novel ANN model, namely Fuzzy Counterpropagation Network (Fuzzy-CPN) has been developed, which is a prototype based mapping network with a self growing structure. It has a fast learning capability in order to perform a continuous function approximation in an adaptive filter. The Counterpropagation Network paradigm is chosen as the basic paradigm due to its features that are simple, fast, and easy to train. The architecture of this proposed model is self-growing. The architecture of the network always adapts during the training process and has three main activities of the fuzzy neurons at the middle layer : neuron generation, neuron adaptation and neuron annihilation.

The constructive learning by specialisation with internal states is applied in this proposed model. It only requires minimal number of iterations. During the learning phase, the Fuzzy-CPN generates the fuzzy neurons from the set of examples of input-output pairs. Each fuzzy neuron performs the fuzzification process and acts as the fuzzy rule.

Although the Fuzzy-CPN is a prototype-based mapping network, during the recall phase it does not produce a prototype as a winner but produces the fit value of the input vector to each prototype. Since there is no mechanism for finding the winning prototype, the recall mechanism can be implemented in parallel processing directly. Furthermore, this proposed model does not suffer from the grandmother cell problem which is usually encountered by the competitive model and becomes more robust than the competitive model.

The Fuzzy-CPN has shown the capability to perform non-linear function approximation by employing the learning mechanism using the input-output pair of the approximated function. The FCPN can produce a discontinuous and non-linear function. The result is promising and shows that the FCPN has good generalisation and localisation properties in approximating a function. This function approximation capability is suitable for the adaptive filter task such as system identification, signal estimation and signal correction in time domain or in frequency domain.

The Fuzzy-CPN has learning parameters that are easy to specified and not sensitive to the network performance. Thus, this Fuzzy-CPN is easy to build and train. To adjust the learning parameter the following steps are applied. Firstly, the number of middle neurons is set to be unlimited. Secondly, ρ is set in the range 0.01-0.02 and the ρ value that produce the lowest MSE with the smallest number of fuzzy neuron is chosen. Thirdly, the number of available resource for the middle layer is reduced to the 50%-60% of the maximum fuzzy neuron number generated for the chosen ρ. Furthermore, to obtain further improvement, the α_{limit} can be set to be equal to 0.6. Since the learning is performed quickly the adjustment of the learning parameter is not time consuming.

The Fuzzy-CPN model has been applied to solve to predict the Mackey-Glass chaotic time series. It is performed by applying the non-linear function approximation capability of the Fuzzy-CPN. In addition, it has been shown that the on-line update capability of this network provides an error reduction in the recall phase. In contrast with another ANN model which requires an iterative learning scheme, the Fuzzy-CPN can implement an on-line adaptation process. The on-line adaptation capability is suitable for the real time applications, especially, in time series

prediction problem, after performing a prediction, the real value of the next time series can be used to refine the network for predicting the future value.

The associative memory capability of the Fuzzy-CPN is applied in a background noise elimination system for the input signal that is corrupted by non-deterministic noise and non-stationary noise. The Fuzzy-CPN model is combined with the spectrogram as time-frequency representation, the periodogram and associative memory for the noise power estimation that are used in the spectra subtraction technique.

By extending the conventional spectrogram method to the proposed spectrogram method, the time-shift variance and time-origin variance of the signal can be tackled by the model. Thus, the proposed spectrogram can extract finer details on the frequency changes of the signal results in better representation on the frequency feature of the signal. The formulation of the proposed spectrogram has been given. By using overlapped blocks of signal has produced the frequency representation of the signal to overcome the boundary problem and provides better noise power spectra estimation.

Although the Fuzzy-CPN is a prototype based mapping network, the final output of the Fuzzy-CPN is not always one member of the prototype sets. This is due to the recall mechanism of the Fuzzy-CPN which employs the fuzzy set approach. The final output may be the combination of the prototypes or a new model of the prototype. The final output itself may be a new form of any member in the prototype set and never exists in the set of prototype. Therefore, the ability of the Fuzzy-CPN in doing novelty detector or to deal with plasticity-stability problem is shown.

Software simulation results with some typical signals corrupted by Gaussian noise are also presented. Followed by the simulation using the real noise which are the street noise, the crowded in a pub and the footstep. After the system has been trained using a particular noise, the sinusoidal signal as the desired signal is mixed with that noise and the system eliminates the noise and reproduces the desired output signal. These simulations show that the proposed noise elimination model can work effectively in the situation where the SNR of the corrupted signal is low. In a low SNR condition, the proposed noise elimination model performs noise power spectra estimation that is used in the spectral subtraction process better than in a high SNR condition. In a high SNR condition, the desired signal becomes dominant compare to the noise, therefore, the noise power spectra estimation becomes more difficult, but in this condition a noise elimination process becomes less important. The simulation shows a significant improvement and a promising result to be applied in the real time application using a dedicated signal processor system.

7.2 Future Improvements

In the implementation of this proposed model there are some constraints . Firstly, since the radial function is used as the basis function, the dimension of the input space determines the generalisation and the localisation of the model. Secondly, since this proposed model performs learning by using the internal state, the amount of memory that is used by the model limits the implementation of the model, because each neuron must store their own internal state.

Further improvements to the model can be performed. They are suggested as follow :

- When the fuzzy neuron performing the expansion of the receptive field in the Step 5. Eq 4.19. *A coefficient of expansion* can be inserted into this function to control that the receptive field will not grow unbounded. This mechanism give more effects to control the expansion of the receptive field of a fuzzy neuron when a fuzzy neuron is just generated.
- *Neuron fusion* mechanism can be incorporated to this model. Instead of only annihilating a fuzzy neuron, the fuzzy neuron that will be annihilated is merged to the closest fuzzy neuron. This mechanism improves the capability of the model to cover the whole input space without forgetting the old patterns.
- In this proposed model, during the learning the receptive of fuzzy neuron is always expanded. It can be improved by *contracted the region of receptive field* which has been used in some fuzzy neural model such as Fuzzy Min-Max (Simpson et al., 199)
- During the training a particular *cost function* can be applied in order to control the neuron generation process or neuron annihilation process. This cost function can be MSE of the prediction output and the desired signal or the separation of the fuzzy partition.

7.3 Conclusions

The formal model of the neuron, fuzzy neuron and the architecture of an ANN model has been described in this thesis. This formulation is used to specify a novel fuzzy neural network architecture which is identified as the Fuzzy-CPN. Although this model is a prototype-based mapping network that employs Counterpropagation paradigm as the basic paradigm, it does not suffer from the grand mother cell

problem and it can perform the novelty detector. This proposed model combines the fuzzy set approach with the artificial neural network techniques, a combination of both techniques may result in a learnable system that can tackle the vagueness problem of a changing environment. The capability to perform a non-linear function approximation is used as the basic structure of the adaptive filter, which can operate in an unknown environment. The Fuzzy-CPN has fast learning capability and self-growing structure, and it applies the constructive learning by specialisation with the internal states. This learning scheme is expected to be able to reduce the development time and cost of the designing adaptive filters based on fuzzy set approach, because it is easy to specify the learning parameter of the Fuzzy-CPN.

The Fuzzy-CPN has been applied to tackle the adaptive filter tasks. It has been applied for the signal estimation problems. This model is used to predict the Mackey-Glass chaotic time series. The result of simulations shows that the Fuzzy-CPN can predict with only a small numbers of learning iteration and produces an acceptable result. For the signal correction tasks, this model has been applied for the background noise elimination problem by combining the Fuzzy-CPN with time-frequency representation and spectral subtraction method. The result is promising to be applied in a dedicated Digital Signal Processing system for real-time applications because this Fuzzy-CPN model can be easily implemented in a parallel processing system.

References

Abu-Mostafa, Yaser S. (1989, November). Information theory, complexity, and neural networks. *IEEE Communication Magazines*, 27, pp. 25-28.

Anderson, James A., Michel T. Gately, P. Andrew Dens, R. Collins (1990). Radar signal categorization using a neural network. *Proceedings of the IEEE*, 78(1), 1646-1657.

Arai, Masahiko (1989). *Mapping abilities of three-layer neural networks*. Paper presented at 1989 IEEE Internatonal Joint Conference on Neural Network, I, 419-423.

Asakawa, Kazuo, Hideyuki Takagi (1994). Neural networks in Japan. *Communication of the ACM*, 37 (3), 106-112.

Back, A. D. , A. C. Tsoi (1991). FIR and IIR Synapses, a new neural network architecture for time series modeling. *Neural Computation*, 37, 375-385.

Barbour, G., K. Blackwell, T. Busse, D. Alkon, T. Vogl (1992). *Dystal: a self-organizing ANN with pattern independent training time.* Paper presented at 1992 IEEE International Joint Conference on Neural Networks, IV, 814-819.

Barnard, Etienne, Ronald A. Cole, Mathew P. Vea, Fileno A. Alleva (1991). Pitch detection with a neural-net classifier. *IEEE Transactions on Signal Processing*, 39 (2), 298-407.

Benaim Michael, Linda Tomasini (1991). Competitive and self-organizing of an information criterion. In Eckmiller, R., G. Hartmann, G. Hauske (Eds.). *Pararel Processing in Neural System Computers* (pp. 391-397). North-Holland: Elsevier Science Publishers B. V.

Berenji, Hamid R., Pratap S. Khedkar (1993). *Clustering in product space for fuzzy inference*. Paper presented at 1993 IEEE Conference on Fuzzy System, 2, 1402-1407.

Betts, Dave, Gordon Reid (1993, March). DSP and audio restoration. *Studio and Sound and Broadcast Engineering*, pp.71-76.

Bezdek, James C., Eric Chen Kuo Tsao, Nikhil R. Pal (1992). *Fuzzy kohonen clustering networks*. Paper presented at 1992 IEEE Conference on Fuzzy System, 1035-1043.

Biglieri, Ezio, Allen Gersho, Richard D. Gitlin, Tong Leong Lim (1984). Adaptive cancelation of nonlinear intersymbol interference for voiceband data transmission. *IEEE Journal on Selected Area in Communications*, 2(5), 765-777.

Blesser, B., J. M. Kates (1). Digital processing in audio signals. In Oppenheimer Alan V. (Ed.). *Application of digital signal processing* (pp.29-116). New Jersey : Prentice-Hall, Inc., Englewood Cliffs.

Boll, S. F. (1979). Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustic, Speech, an Signal*, 27, 113-120.

Boll, Steven F. (1979). Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-27, 113-120.

Boll, Steven F. (1983). Speech enhancement in the 1980s: noise suppresion with pattern matching.(pp.309-323).

Botros, Nazeih M., S. Premnath (1992). *Speech recognition using dynamic neural networks*. Paper presented at 1992 IEEE International Joint Conference on Neural Networks, IV, 737-742.

Bottou, Leon, and Patrick Gallinari (1992). *A unified formalism for neural net training algorithms*. Paper presented at 1992 IEEE International Joint Conference on Neural Networks, IV, 7-12.

Brandt, Michael E. (1991, May). Comparing signals in the time domain. *The C Users Journal*, pp. 58-53.

Carpenter, Gail A., Stephen Grossberg (1988, March). The ART of adaptive pattern recognition by a self-organizing neural network. *Computer*, pp. 77-88.

Carpenter, Gail A., Stephen Grossberg, Natalya Markozon, John H. Reynolds, David B. Rosen (1992). Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3(5), 698-713.

Casselman, F. K., D. F. Freeman, D. A. Kerrigan, S. E. Lane, Millstrom, W. G. Nichols Jr. (1991). *A neural network-based passive sonar detection and classification design with a low false alarm rate.* Paper presented at IEEE Conference on Neural Networks for Ocean Engineering, 49-55.

Caudill, M. (1988). *Network paradigm selection guidelines for application developments.* Paper presented at Proceeding of the Fourth Annual Artificial Intelligence and Advanced Computer Technology Conference, 298-302.

Caudill, Maureen, (1992, June). The view from now. *AI Expert,* pp. 24-31.

Cheng, Yan Ming, Douglas O'Shaughness (1991). Speech enhancement based conceptually on auditory evidence. *IEEE Transactions on Signal Processing*, 39 (9), 1943-1954.

Chow, T. W. S, and Yam, Y. F. (1990). *Discrete Time Domain Poles Zeros Identification using Back Propagation Neural Networks.* Paper presented at International Symposium on Signal Processing and Its Applications : ISSPA '90 : Gold Goast Australia, August 27-31, 1990,

Classen, T. A. C. M. and W. F. G. Mecklenbrauker (1983). Overview of adaptive techniques in signal processing, In Schussler H. W. (Ed). *Signal processing II: Theories and Application* (pp. 747-754). North-Holland: Elsevier Science Publisher B. V.

Cohen. L (1992). Introduction: A Primer on Time-Frequency Analysis. In Boashash. B (Ed.). *Time Frequency Analysis, Methods and Applications* (pp.17-73). New York: Longmann.

Cohen, Leon , Chongmoon Lee (1990). *Optimal windows for the short-time fourier transform*. Paper presented at ISSPA'90 Signal Processing, Theories, Implementation and Applications, Gold Goast, Australia 27-31 August 1990, 912-917.

Cohen, L. Chongmoon Lee (1992). Instantaneous Bandwidth, In Boashash. B (Ed.). *Time Frequency Analysis, Methods and Applications* (pp. 98-117). New York: Longmann.

Cohen, Michael A., Stephen Grossberg, Lonce Wyse (1992). *A neural network for synthesizing the pitch of an acoustic source*. Paper presented at 1992 IEEE International Joint Conference on Neural Network, IV, 649-654.

Cohen, M. E., D. L. Hudson (1992). *Approaches to handling of fuzzy input data in neural networks.* Paper presented at 1992 IEEE Conference on Fuzzy System, 93-100.

Coley, J. W., J. W. Tukey (1965). An algorithm for the machine computation of complex Fourier series. *Math. Comp.*, 19, 297-301.

Coloma, J., I.R. Carden, R.A. Carraso (1991). Adaptive artificial neural network algorithms for MCD applications. *IEE Colloqium on Adaptive Filtering Non Linear Dynamics and Neural Networks*, 9, 1-8.

Cohen, M. H., P. O. Pouliquen, A. G. Andreou (1991). *An Auto-adaptive synthetic neural network for real-time separation of independent signal sources.* Paper presented at IJCNN-91-Seattle; International Joint Conference on Neural Network, July 8-12, 1991, Washington State Conention & Trade Centre, Seattle WA, 1, 211-214.

Connel, J. M., and C. S. Xydeas (1990). *A comparison of acoustic noise cancellation techniques for telephone speech.* Paper presented at Paper presented at International Symposium on Signal Processing and Its Applications : ISSPA'90 : Gold Goast Australia, August 27-31, 1990,

Cotter, Neil E. (1990). The Stone-Weierstrass theorem and its application to neural networks. *IEEE Transactions on Neural Networks*, 1 (4), 290-295.

Cox, Earl (1992, January). The great myths of fuzzy logic. *AI Expert* , pp. 41-45.

Cox, Earl (1992, June). Integrating fuzzzy logic into neural nets. *AI Expert*, pp. 43-47.

Currie, M. G. (1992). *An optimized architecture incorporating a neural net.* Paper presented at 1992 IEEE International Joint Conference on Neural Networks, IV, 543-548.

Cybenko, G. (1990). Complexity theory of neural networks and classification problems. In Almeida L.B., and C. J. Wellekens (Eds.). *Neural Networks, EURASIP Workshop 1990* (pp. 27-44). London:Springer-Verlag.

Dasarathy, Belur V. (1992). *FLUTE : Fuzzy learning in unfamiliar teacher environments.* Paper presented at 1992 IEEE Conference on Fuzzy System, 1070-1077.

Dave, Rajesh N., Kurra Bhaswan (1992). Adaptive fuzzy c-shells clustering and detection of ellipses. *IEEE Transactions on Neural Networks*, 3(5), 643-662.

Deller, John R., John G. Proakis, John H. L. Hansen (1992). *Discrete-time processing of speech signal.* USA : Macmillan Publishing Company.

Momentum Data System, (1992). DPSWorks for Windows Version 2.0, Digital Signal Processing Application.CA

Dubois, Didier, Henri Prade (1993). *Fuzzy sets and probability: misunderstanding, bridges, and gaps.* Paper presented at 1993 IEEE Conference on Fuzzy Systems, 2, 1059-1068.

Duhamel, Pierre (1986). Implementation of "Split-Radix" FFT algortihms for complex, real, and real-symmetric data. *IEEE Transactions on Acoustic, Speech, and Signal Processing*, ASSP-34 (2), 285-295.

Duhamel, Pierr (1990). Algorithms meeting the lower bounds on the multiplicative of length-2^n DFT's and their connection with practical algortihms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28 (9), 1504-1511.

Duhamel, P., B. Piron, J. M. Etcheto. (1988). On computing the inverse DFT. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36 (2), 285-286.

Duhamel, Pierre, Martin Vetterli (1987). Improved Fourier and Hartley transform algorithms: application to cyclic convolution of real data. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35 (6), 818-824.

Eisenberg, Joe, Feld, David, Edwin Lewis (1988). A passive shared element analog electrical cochlea. In David, S. (Ed). *Advanced in Neural Information Systems* (pp. 663-669). San Mateo: Morgan Kuffman Publisher.

Evans, David M. W. (1987). An improved digit-reversal permutation algorithm for the Fast Fourier and Hartley transforms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35 (8), 1120-1125.

Fahlman Scott E., Christian Lebiere (1990). The cascade-correlation learning architecture. In Touretzky (Ed.). *Advances in Neural Information Processing Systems 2* (pp. 524-532). San Mateo: Morgan Kauffman Publishers Inc.

FAQ, (1994). *Frequently Asked Question of Fuzzy Logic*, USENET.

Farley, James F., Peter D. Warhol (1993, February). Neural nets for predicting behaviour. *Dr Dobb Journal*, pp. 82 - 85.

Farmer and Sidorowich (1987). Prediciting chaotic time series. *Physical Review Letters*, 59, 845.

Finster, Herald (1992). *Automatic speech segmentation using neural network and phonetic transcription*. Paper presented at 1992 IEEE International Joint Conference on Neural Networks, IV, 734-736.

Flanagan, James (1972). *Speech Analysis Synthesis and Perception* (2nd ed.). New York: Springer-Verlag.

Frizke, Bernd (1993). Kohonen feature maps and growing cell structures a performance comparison. In Gilles, C. L., S. J. Hanson, J. D. Cowan (Eds.). *Advances in Neural Information Processing System*. San Mateo: Morgan Kaufmann Publisher.

Frizke, Bernd, (1993). *Vector quantization with a growing and splitting elastic net*. Paper presented at ICANN '93: Proceedings of the International Conference on Artificial Neural Networks, 13-16.

Furukawa, Tokoshiro and Hajime Kubota (1990). *A new communication processing system with lower error-rate and high reliability*. Paper presented at International Symposium on Signal Processing and Its Applications : ISSPA '90 : Gold Goast Australia, August 27-31, 728-731.

Gee, S., M. Rupp (1991). *A comparison of adaptive IIR echo canceler*. Paper presented at ICASSP-91 International Conference on Acoustics Speech and Signal Processing, 3, 1541-1544.

Gerald, J. A. B., N. L. Esteves, M. M. Silva (1990). *An adaptive IIR echo canceller combining output error and equation error*. Paper presented at 1990 IEEE International Symposium on Circuits and Systems, 1, 779-783.

Gersho, A. (1984). Adaptive filtering with binary reinforcement. *IEEE Transactions of Information Theory*, IT-30, 191-199.

Ghosh, Ashish, Nikhil R. Pal, Sankar K. Pal (1993). Self-organization for object etraction using a multilayer network and fuzziness measures. *IEEE Transactions on Fuzzy System*, 1(1), 54-68.

Gibson, Gavin J., Colin F. N. Cowan (1990). On the decision regions of multilayer perceptrons. *Proceedings of the IEEE*, 78 (10), 1590-1594.

Godwin, W. H., M. J. Heering, D. P. Goodall, W. Bates (1988). Identication of speech in noise using low-cost algorithms. In Moscardidi, A. O., E. H. Robson (Eds.). *Mathematical modelling for information technology : Telecommunication transmission, reception and security* (pp.42-49). Great Britain: Ellis Horwood Limited.

Gorman, R. P., Sejnowski, T. J. (1988). Analysis of hidden units in a hidden layered network trained to classify sonar targets. *Neural Networks*, 1(1), 75-89.

Grifftihs, L. J., (1967). A simple adaptive algortihm for real time processing in antenna arrays. *Proceedings of the IEEE*, 57, 1696-1704.

Hambaba, Mohamed L., Ali Charchali (1990). *Unsupervised learning the hidden structure of speech*. Paper presented at The 22nd South Eastern Symposium

on System Theory, March 11-13, 1990, Tennesse TEchnological University, Cookevulle, Tennesse, 662-666.

Hansen, John H. L., Mark A. Clements (1991). Constrained iterative speech enhancement with application to speech recognition. *IEEE Transactions on Signal Processing*, 39 (4), 795-805.

Hayashi Yoichi, James J. Buckley, Ernest Czogala (1993). Fuzzy neural network with fuzzy signals and weights. *International Journal of Intelligent System*, 8, 527-537.

Hecht-Nielsen, Robert (1987a). Counter Propagation Network. *Applied Optics*, 26(23), 4979-4984.

Hecht-Nielsen, Robeter (1987b). *Kolmogorov's mapping neural network existence network*. Paper presented at the IEEE First International Conference on Nueral Network - San Diego, III, 11-14.

Hecht-Nielsen, Robert (1988a). Neurocomputing: picking the human brain. *IEEE Spectrum*, 25(3), 36-41.

Hecht-Nielsen, Robert (1988b). Applications of counterpropagation networks. *Neural Networks*, 1, 131-139

Hecht-Nielsen, Robert (1990). Neurocomputing. USA : Addison-Wesley Publishing Company, Inc.

Hewitt, P. D., P. J. C. Skitt, R. C. Witcomb (1989). A self-organizing feedforward network applied to acoustic data. In Taylor John, C. L. T. Mannion (Eds.). *New developments in neural computing* (pp. 71-78). England : IOP Publishing Ltd.

Horvath, S. (1983). Digital Signal Processing in communication and transmission. In Schüssler (Ed.) *SIGNAL PROCESSING II: Theories and Applications* (pp. 515-522). North-Holland: Elsevier Science Publisher B. V.

Howell, J. A. C. W. Barnes, S. K. Brown, G. W. Flake, R. D. Jones, Y.C. Lee, S. Qian, and R. M. Wright (1989). *Control of a negative-ion accelreator source*

using neural networks. Paper presented at the International Conference on Accelerator and Large Experimental Physics Control Systems.

Hoyt, J. D., H. Wechster (1990). *An examination of the application of multilayer neural-networks to audio signal processing.* Paper presented at IJCNN: International Joint Conference on Neural networks, January 15-19, 1990, Omni Shoreham Hotel, Washington DC, 305-310.

Iwata, Akira, Yoshihisa Suwa, Yutaka Ino., Nobuo Suzumura (1992). *Hand written alpha-numeric recognition by a self-growing neural network "CombNet-II".* Paper presented at 1992 IEEE International Joint conference on Neural Networks, IV, 228-234.

Jang, J-S Roger (1992). *Fuzzy controller design without domain experts.* Paper presented at 1992 IEEE Conference on Fuzzy System, 289-296.

Jang J-S Roger, Chuen-Tsai Sum (1993). *Predicting Chaotic Time Series with Fuzzy If-Then Rule.* Paper presented at 1993 IEEE Conference on Fuzzy System, 1079-1084.

Jeffries D. J., D. R. Farrier (1990). Multiple signal detection with unknown background noise. 657. In Durrani, T. S., J. B. Abbis, J. E. Hudson, R. N. Madan, J. G. McWhirter, T. A. Moore (Eds.). *Mathematcis in signal processing* (pp. 657-667). Oxford : Clarendon Press.

Jianxin Jiang, Hu Zheng, Liu Feng (1992). *A hybrid neural fuzzy neural framework for speech recognition.* Paper presented at 1992 IEEE International Joint Conference on Neural Networks, IV, 643-648.

Jones, R. D., W. C. Mead, Y. C. Lee, C. W. Barnes, G. W. Flake, L. A. Lee, M. K. O'Rourke (1991). *Using CNLS-Net to predict the Mackey-Glass chaotic time series.* Paper presented at IJCNN-91-Seattle, International Joint Conference on Neural Network, July 8-12, 1991, Washingto State Convention & Trade Center, Seattle, WA, II, 485-490.

Jones, R. R., Y. C. Lee, C. W. Barnes, G. W. Flake, K. Lee, P. S. Lewis, S. Qian (1989). *Function approximation and time series prediction with neural*

networks. Paper presented at International Joint Conference on Neural Network, Sheratorn Washington Hotel, 1989,I, 649-665.

Kadambe, Shubha, G. Faye Boudreaux-Bartels (1992). Application of the wavelet transform for pitch detection of speech signals. *IEEE Transactions on Information Theory*, 38 (2), 917-924.

Kailath, L. (1980). *Linear systems*. NJ : Prentice-Hall, Englewood CLiffs.

Kalman, Barry L., and Stan C. Kwasny (1992). *Why Tanh: choosing a sigmoidal function*. Paper presented at 1992 IEEE International Joint Conference on Neural Networks, IV, 578-581.

Kanekar, Ashish, J, and Ali Feliachi (1990). *State Estimation Using Artificial Neural Networks*. Paper presented at 22nd South Eastern Symposium on System Theory, March 11-13, 1990, Tennesse Technological University, Cookeville, Tennesse, 552- 554.

Kaufmann, A., M. M. Gupta (1988). *Fuzzy Mathematical Models in Engineering and Management Science*. North Holland : Elsevier Science Publsiher B. V.

Kipersztok, Oscar (1993). *Active control of broadband noise using fuzzy logic*. Paper presented at 1993 IEEE Conference on Fuzzy Systems, II, 906-911.

Klirr, Goerge J., Tina A. Folger (1988). *Fuzzy sets, uncertainty and information*.NJ: Prentice Hall, Englewood Cliffs.

Kobatake, Hidefumi and Kaoru Gyoutoku (1990). *Restoration of speech Contamined by Nonstationary Noises*. Paper presented at International Symposium on Signal Processing and Its Applications : ISSPA '90 : Gold Goast Australia, August 27-31, 495-498.

Kohonen, T. (1990). The Self-Organizing Map. *Proceedings of the IEEE*, 78(9), 1461-1480.

Kosko, Bart (1986). Fuzzy entropy and conditioning. *Information Sciences*, 40, 165-174.

Kosko, Bart (1987). Fuzzy Associative Memory. In A. Kendel (Ed.). *Fuzzy Expert Systems*. MA: Addison-Wesley.

Kosko, Bart (1990). Unsupervised learning in noise. *IEEE Transactions on Neural Networks*, 1(1), 44-57.

Kosko, Bart (1992a). *Fuzzy system as universal approximators.* Paper presented at 1992 IEEE Conference on Fuzzy System, 1153-1162.

Kosko, Bart (1992b). *Neural Networks and Fuzzy Systems, A dynamical systems approach to machine intelligence*. NJ : Prentice-Hall, Engelwood Cliffs.

Kosko, Bart (1993). *Fuzzy Thinking, The New Science of Fuzzy Logic.* USA: Harper Collings Publisher.

Kosko, Bart, Julie Dickerson (1994, July). Fuzzy virtual worlds. *AI Expert*, pp.25-31.

Kowalczyk, Adam (1991). *Can multilayer mapping networks with finite number of real parameters harness the computational power of Komogorov's theorem ?.* Paper presented at 1991 IEEE International Joint Conference on Neural Networks, The Westing Stamford and Westin Plaza, 18-21 November 1991, Singapore, 3, 2722-2728.

Krishnapuram Raghu, Olfa Nasraoui, Hichem Frigui (1992). The fuzzy C spherical shells algorithm: a new approach. *IEEE Transactions on Neural Networks*, 3(5), 663-671.

Kuo S., H. Shao (1990). *A real time acoustics echo cancellation systems.* Paper presented at IEEE International Conference on System Engineering, 168-171.

Kuo, S. M., J. Chen (1992). New adaptive IIR notch filter and its application to howling control in speaker phone system. *Electronics Letters*, 28(8), 764-766.

Kwan, Hon-Keung, Pang-Chung Tsang (1990). *Systolic implementation of counterpropagation networks.* Paper presented at ICASSP '90 1990 International Conference on Acoustics, Speech and Signal Processing, 2, 953-956.

Lee, Chuan Chien (1990). Fuzzy-Logic in control systems: Fuzzy logic controller part-I. *IEEE Transactions on System, Man, and Cybernetics*, 20 (2), 404-418.

Lee, Chuan Chien (1990). Fuzzy-Logic in control systems: Fuzzy logic controller part-II. *IEEE Transactions on System, Man, and Cybernetics*, 20 (2), 419-435.

Lee, Tsu-Chang (1991). *Structure Level Adaptation for Artificial Neural Network*. London: Kluwer Academy Publisher.

Lee, Samuel C., Edward T. Lee (1975). Fuzzy neural networks. *Mathematical Biosciences*, 23, 151-177.

Leemon, Michael (1991). *Competitively Inhibited Neural Networks for Adaptive Parameter Estimation*. London : Kluwer Academic Publishers.

Levin, Esther, Naftali Tishby, Sara A. Solla (1990). A statistical approach to learning and generalization in layered neural networks. *Proceedings of the IEEE*, 78 (10), 1568-1574.

Lim Kyoung-Man, Young-Chui Sim, Kyung-Whan Oh (1992). *A face recginition system using fuzzy logic and artificial neural network*. Paper presented at 1992 IEEE Conference on Fuzzy System, 1063- 1069.

Lin, Z. K. Khorasani, R. V. Patel (1990). *A counterpropagation neural networks for function approximation*. Paper presented at 1990 IEEE International Conference on Systems, Man and Cybernetics, 382-384.

Lippman, Richard P. (1987, April). An introduction to computing with neural nets. *IEEE ASSP Magazine*, pp. 4-22.

Lippmann, Richard, Paul Beckman (1989). Adaptive neural net preprocessing for signal detection in non-gaussian noise. In Touretzky (Ed.). *Advances in neural information processing systems*, I, (pp. 124-132). San Mateo : Morgan Kaufmann Publishers Inc.

Liu, Jun, Danwei Wang (1992). *Data compression for image recognition using neural network*. Paper presented at 1992 IEEE International Joint Conference on Neural Networks, IV, 333-338.

Long, G. F. Ling (1990). *A three complex system identification method and its application to echo canceller on fast initialization*. Paper presented at

ICASSP'90 - 1990 International Conference on Acoustics, Speech, Signal Processing, 3, 1671-1674.

Machado Ricardo Jose, Armando Freitas da Rocha (1992). *Evolutive fuzzy neural networks.* Paper presented at 1992 IEEE Conference on Fuzzy System, 493-500.

Magotra, N., D. Hush, J. Bibbo, E. Choel (1991). *Seismic Signal Discrimination using Adaptive System Parameters.* Paper presented at Proceeding of the 33rd Midwest Symposium on Circuits and Systems, 84-87.

Malkoff, Donald B. (1992). Detection and classification by neural networks and time-frequency distributions. In Boashash, B. (Ed.). *Time Frequency Analysis, Methods and Applications* (pp. 423-348). New York: Longmann.

Malvar, Henrique S. (1992). *Signal Processing with Lapped Transforms.* Norwood: Artech House Inc.

Manabe, Takeshi, Ryuji Kaneda, (1991). *Adaptive decision-feedback of digital transmission channels using forward only counterpropagation networks.* Paper presented at 1991 IEEE International Joint Conference on Neural Networks, 1, 220-225.

Marcos, S., O. Macchi, C. Vignal, G. Dreyfus, L. Perzonnaz, P. Roussel-Ragot (1992). A unified framework for gradient algorithms used for filter adaptation and neural network training. *International Journal of Circuit, Theory, and Application,* 20(2), 159-200.

Matsuyama Yasuo (1992). *Learning in competitive networks with penalties.* Paper presented at 1992 IEEE International Joint Conference on Neural Networks, IV, 773-778.

Matthews, M. B. (1990). Nonlinear adaptive filtering using neural networks. *Mitteilungen AGEN,* 51(special), 13-23.

Matthews, Michael B., G.S. Moschytz (1990). *Neural network nonlinear adaptive filtering using extended Kalman filter algorithm.* Paper presented at International Neural Network Conference 1990, 115-118.

Mc Neill, Daniel, Paul Freiberger (1993). *Fuzzly logic, the discovery of a revolutionary computer technology and how it is changing our world.* Melbourne : Bookman Press.

Mehra, Pankaj, Benjamin W. Wah (1992). *Artificial Neural Networks : Coneepts and Theory.* CA: IEEE Computer Society Press.

Messerschmitt David, David Hedberg, Christopher Cole, Amine Haoui, Peter Winship (1986). Digital voice echo canceller with a TMS32020. *Digital Signal Processing Applications with TMS320 Family* (pp.415-454). Texas Instruments.

Mitra Sushmita, Sankar, K. Pal (1994). Logical operation based fuzzy MLP for clasification and rule generation. *Neural Networks*, 7(2), 353-373.

Moddy J., C. J. Darken (1989). Fast learning in networks of locally tuned processing unit. *Neural Computation*, 1, 281-294.

Munakata, Toshinori, Yashvant Jani (1994). Fuzzy systems: an overview. *Communication of the ACM*, 37 (3), 69-76.

Murata Noboru, Shuji Yoshizawa, Shun-ichi Amari (1992). Network information criterion determining the number of hidden units for an artificial neural network model. *Research Report*, University of Tokyo, June 22 1992.

Murphy, Owen J. (1990). Nearest neighbor pattern classfication. *Proceedings of the IEEE*, 78 (10), 1595-1598.

Mutluay, H. E., M. M. Fahmy (1984). Adjustable digital filter. In V. Cappelini, A. G. Constantinides (Eds.). *Digital Signal Procesing-84* (pp.165-160). North-Holland: Elsevier Science Publishers B. V.

Naik, K., G. Singh, K. Khorasani, R. V. Patel (1992). *An improved multidirectional associative memory.* Paper presented at 1992 IEEE International Joint Conference on Neural Networks, 1506-1511.

Nerrand, O., P. Roussel-Ragot, L. Personnaz, G. Dreyfus (1993). Neural networks and non linear adaptive filtering: unifying concepts and new algorithms. *Neural Computation*, 5, 165-199.

Newton, Scott C., Surya Pemmaraju, Sunanda Mitra (1992). Adaptive fuzzy leader clustering of complex data sets in pattern recognition. *IEEE Transactions on Neural Networks*, 3(5), 794-800.

Nguyen D., B. Widrow (1989). *The truck backer-upper: An example of self-learning in neural networks.* Paper presented at 1989 IJCNN International Joint Conference on Neural Networks, Sheratorn Washington Hotel, II, 357-367.

Nie, Junhong, D. A. Linkens (1993). Learning control using fuzzified self-organizing radial basis function network. *IEEE Transactions on Fuzzy Systems*, 1(4), 280-287.

Niranjan, Mahesan, Frank Fallside (1990). Speech feature extraction using neural networks. In Ameida, L. B., C. J. Wellekens (Eds.). *EURASIP Workshop 1990, Sesimbra, Portugal, February 15-17* (pp.197-204). Berlin : Springer-Verlag.

Nolfi, Stefano, Domenico Parisi (1992). Growing neural networks, Departement of Cognetive Processes and Artificial Intelligence, Italy. *Technical Report.*

Oppenheimer, Alan V., Ronald W. Schafer (1989). *Discrete-Time Signal Processing.* NJ : Prentice Hall Inc.

Palakal, Mathew J., Michael J. Zoran (1992). *A neural network approach to large dimensional spectral pattern processing.* Paper presented at 1992 IEEE International Joint Conference on Neural Networks, IV, 691-695.

Palakan, M. J., M. J. Zoran (1989). Feature extraction from speech spectrogram using multi-layered network model. *IEEE International Workshop on Tools for Artificial Intelligence. Architectures, Languages and Algortihms*, 224-230.

Pao,Yoh-Han (1989). *Adaptive pattern recognition and neural networks.* USA: Addison-Wesley Publishing Company.

Park, Y. H., S. Y. Bang (1991). *A new neural network model based on neighbor classifier.* Paper presented at 1991 IEEE International Joint Conference on

Neural Networks, The Westing Stamford and Westin Plaza, 18-21 November 1991, 3, 2386-2389.

Pitas Ioannis, Evangelos Milios, Anastasios N. Venetsanopoulos (1992). A minimum entropy approach to rule learning from examples. *1992 IEEE Transactions on System , Man and Cybernetics*, 22 (4), 621-635.

Poggio, Tomaso, Federico Girosi (1990). Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), 1461-1497.

Qureshi, Shahid H (1985). Adaptive Equalization. *Proceedings of the IEEE*, 73(9), 1349-1387.

Radivojevic, I., Jayantha Herath, W. Steven Grary. (1991, June). High performance DSP architecture for intelligence and control applications. *IEEE Control System Magazine*, 11(4), 49-55.

Refenes, A. N., S. Vithalani (1991). Constructive learning by specialisation. In R. G. Hartmann, G. Hauske (Eds.). *Pararell Processing in Neural System Computers* (pp. 923-928). North-Holland: Eisevier Science Publishers B. V.

Rock, Denny, Don Malkoff, Ron Stewart (1993, February). AI and aircraft healt monitoring. *AI Expert*, pp.28-35.

Roger Jang, Jyh-Shing (1993). ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Transactions on Systems, Man, and Cybernetics*, 23 (3), 665-685.

Roitblat, H. L., P. W. B. Moore, P. E. Nachtigall, R. H. Penner, W. W. L. Au (1989). *Dolphin echolocation: identification of returning echoes using a couinterpropagation network*. Paper presented at 1989 IJCNN International Joint Conference on Neural Networks, Sheraton Washington DC, 1, 295-300.

Rumelhart, D. E., D. Zipser (1985). Feature discovery by competitive learning. In Rumelhart, David E., James L, McClelland (Eds.). *Parallel Distributed Processing, Exploration in the Microstructure of Cognition*, vol I, (pp. 151-193). London : The MIT Press.

Rumelhart D. E., G. E. Hinton, R. J. Williams (1986). Learning internal representations by error propagation. In Rumelhart, David E., James L, McClelland (Eds.). *Parallel Distributed Processing, Exploration in the Microstructure of Cognition*, vol I, (pp. 318-362). London : The MIT Press.

Rumelhart, D. E., G. E. Hinton, J. L. MCClelland (1986). A general framework for parallel distributed processing. In Rumelhart, David E., James L, McClelland (Eds.). *Parallel Distributed Processing, Exploration in the Microstructure of Cognition*, vol I, (pp. 46-109). London : The MIT Press.

Scharf, Louis L. (1991). *Statistical signal processing* (pp.1-4). USA: Addison-Wesley Publishing Company Inc.

Sethi, Ishwar K. (1990). Entropy nets: from decision trees to neural networks. *Proceedings of the IEEE*, 78 (10), 1605-1613.

Shaudys, Fred E., Todd K. Leen (1992). *Feature selection for improved classification*. Paper presented at 1992 IEEE International Joint Conference on Neural Networks, IV, 607-702.

Shimokoshi, Kiyoshi, Yukinao Hashitsuma (1989). *A study of voice/non-voice discrimination method using neural networks for integrated packet switching systems*. Paper presented at 1989 ISCAS, 2096-2099.

Shynk, J. J. (1992, January). Frequency-domain and multirate adaptive filtering. *IEEE Signal Processing Magazine*, pp.14-37.

Simpson, Patrick K. (1990) *Artificial Neural Systems. Foundations, paradigms, applications, and implementations*. USA : Pergamon Press Inc.

Simpson, Patrick, K. (1992). Fuzzy min-max neural networks-part 1: classification. *IEEE Transactions on Neural Networks*, 3(5), 776-786.

Simpson, Patrick K., (1993). Fuzzy min-max neural networks-part 2: clustering, *IEEE Transactions on Fuzzy Systems*, 1(1), 32- 45.

Soleit, E. A., O. R. Hintin, E. Horne (1988). *A new adaptive recursive digital echo canceller for baseband transmission channel*. Paper presented at Proceeding of the IASTED International Symposium Applied Informatics - AI'88, 67-69.

Srikanth, Radhakrishnan, Frederick E. Petry, Cris Koutsougeras (1993). *Fuzzy elastic clustering*. Paper presented at 1993 IEEE Conference on Fuzzy System, 2, 1179-1182.

Srirengan, Swarmala, Chee-Kit Looi (1991). *On using backpropagation for prediction: an empirical study*. Paper presented at 1991 IEEE International Joint Conference on Neural Networks, The Westing Stamford and Westin Plaze, 18-21 November 1991, 1285-1289.

Steam, Samuel D. (1988). Fundamentals of Adaptive signal processing. In Lim, Jae S, Alan V. Openheim (Eds.). *Advanced Topic in Signal Processing* (pp. 246-288). New Jersey: Prentice Hall.

Sugeno, Michio, Takahiro Yasukawa (1993). A fuzzy-logic-based approach to qualitative modelling. *IEEE Transactions on Fuzzy System*, 1 (1), 7-30.

Sztipanovitz, Janos (1990). Adaptive processing with neural network controlled resonator-banks. *IEEE Transactions on Circuits and Systems*, 37(11), 1436-1440.

Taber, Rod (1994, July). Fuzzy Cognitive Maps modes society systems. *AI Expert*, pp.19-23.

Thomopoulos, Stelios C. A., Dimitrios K. Bougoulias (1991). *DIGNET: a self-organizing neural network for automatic pattern recognition and classification*. Paper presented at 1991 IEEE International Joint Conference on Neural Networks, The Westing Stamford and Westin Plaze, 18-21 November 1991, 3, 2683-2692.

Tomlinson. R. W. , W. Treurniet (1990). Spectral processing of harmonic complex tones and pitch by pdp networks. In Eckmiller, R., G. Hartmann, G. Hauske (Eds.). *Parallel processing in neural systems and computers* (pp. 399-403). Norht-Holand : Elsevier Science Publishers B. V.

Tong, Lang, Ruey-wen Liu, Victor C. Soon, Yih-Fang Huang (1991). Indeterminacy and identifiability of blind identification. *IEEE Transactions on Circuits and Systems*, 38 (5), 499-509.

Treichler, John R., C. Richard Johnson JR, Michael G. Larimore (1986). *Theory and design of adaptive filters.* USA: John Willey and Sons.

Trompf, Michael (1992). *Neural network development for noise reduction in robust speech recognition.* Paper presented at 1992 IEEE International Joint Conference on Neural Network, IV, 722-727.

Tsoi, A. C., D. S. C. So, A. Sergejew (1993, March). *Classification of electroencephalogram using artificial neural networks.*

Ulug, M. E. (1992). *A rapid learning orthonormal neural network for signal processing.* Paper presented at 1992 IEEE International Joint Conference on Neural Network, IV, 265-270.

Uncini, A., M . Marchesi, G. Orlandi, F. Piazza (1990). *An adaptive neural networks filters for evoked potentials.* 1990 IEEE International Symposium on Circuits and Systems, 2, 1086-1089.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM,* 27(11), 1134-1142.

Vary, Peter (1983). On the enhancement of noisy speech. In H. W. Schussler (Ed.). *SIGNAL PROCESSING II: Theories and Application* (pp. 327-330). North-Holland: Elsevier Science Publisher B. V.

Vassiliadis, C. A. (1990). *Neural networks - twelve learning algorithms.* Paper presented at the 22nd Southeastern Symposium on System Theory, March 11-13, 1990, Tennessee Technological University, Cookeville, Tennessee, 449-454.

Venkatesh, Santosh S. (1992). *Computation and learning in the context of network capacity.* In Weshster, Harry (Ed.). Neural Networks for Perception: Computation, Learning, and Architecture (vol. 2) (pp. 173-207). United Kingdom: Academic Press Inc.

Wang, Li-Xin, (1992). *Fuzzy system are universal approximators.* Paper presented at 1992 IEEE Conference on Fuzzy System, 1164-1170.

Wang, Li-Xin, Jerry M. Mendel (1992a). Fuzzy basis function, universal approximation, and orthogonal least-squares learning. *IEEE Transactions on Neural Networks*, 3(5), 807-814.

Wang, Li-Xin, Jerry M. Mendel (1992b). Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6), 1414- 1427.

Wang, Li-Xin, Jerry M. Mendel (1993a). Fuzzy adaptive filters, with application to nonlinear channel equalization. *IEEE Transactions on Fuzzy Systems*, 1(3), 161-170.

Wang, Li-Xin, Jerry M. Mendel (1993b). *An RLS fuzzy adaptive filter with application to nonlinear channel equalization.* Paper presented at 1993 IEEE International Conference on Fuzzy System, 2, 895-900.

Wasserman, Philip (1993). *Advanced Methods in Neural Computing.* New York: Van Nostrand Reinhold.

Weber, Mark, Paul B. Crilly (1991). Adaptive Noise Filtering using an Error-Backpropagation Neural Network. *IEEE Transactions on Instrumentation and Measurement*, 40(5), 820-825.

Webos, Paul J. (1991). Links between artificial neural networks (ANN) and statistical pattern recognition. In Sethi, I. K., A.K. Jain (Eds.). *Artificial Neural Networks and Statistical Pattern Recognition - Old and New Connections* (pp. 11-31). North-Holland: Elsevier Science Publisher B. V.

Weibel, A., Hanazawa, T., Hinton, G. Shikano, K, and Lang, K. J, (1989). Phoneme recognition using time-delay neural networks, *IEEE ASSP*, 37, 328-339.

Widrow, B., M. E. Hoff, Jr. (1960). Adaptive switching circuits. *1960 IRE WESCON Convention Record*, part 4, pp. 96-104.

Widrow, B., S. D. Stearn (1985). *Adaptive Signal Processing.* New Jersey : Pretice-Hall.

Wright, J. B., J. B. Foley (1990). Adaptive periodics noise cancellation for the control of acoustic howling. *Signal Processing V : Theories and Applications*

Proceedins of EUSIPCO-90. Fifth European Signal Processing Conference, 3, 1979-1982.

Xue, Q. Y. Hu, W.J. Thomas (1992). Neural Network-based Adaptive Matched Filtering for QRS Detection. *IEEE Transactions on Biomedical Engineering*, 39(4), 317-329.

Yamakawa Takeshi, Masuo Furukawa (1992). *A design algorithm of membership functions for A fuzzy neuron using example-based learning*. Paper presented at 1992 IEEE Conference on Fuzzy System, 75-82.

Zadeh, L. A. (1965). Fuzzy sets. *Information Control*, 8, 338-353.

Zadeh, Lotfi A (1992, March). The calculus of fuzzy logic IF/THEN rules. *AI Expert*, pp. 23-27.

Zadeh, Lotfi A. (1994). Fuzzy logic, neural networks, and soft computing. *Communication of the ACM*, 37 (3), 77-84.

Zhou, Li, Di Tang Fang (1988). *The research on speech feature representaation method and distance measure method.* Paper presented at Proceeding of the Fourth Annual Artificial Intelligence and Advanced Computer Technology Conference, 631-633.

Appendix A

Glossary of Acronyms

AMUSE	Algorithm for Multiple Unknown Signal Extraction
ANC	Adaptive Noise Cancelling
ANFIS	Adaptive Network based Fuzzy Inference System
ANN	Artificial Neural Network
ANVC	Adaptive Noise and Vibration Control
APNC	Adaptive Periodic Noise Cancellation
ARMA	Auto Regressive Moving Average
ART	Adaptive Resonance Theory
BDN	Bi-Directional Neurons
BP	Backpropagation Network
CPN	Counter Propagation Network
CS	Composite State
DCN	Defuzzifier Control Neuron
DFT	Discrete Fourier Transform
DN	Defuzzifier Neuron
EFOBI	Extended Fourth Order Blind Identification
FBF	Fuzzy Basis Function
FBP	Feed Back Path
FCPN	Fuzzy Counter Propagation Network
FFP	Feed Forward Path
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FN	Fuzzy Neuron
FS	Fuzzy System
GRNN	General Regression Neural Network
ICM	Input Connection Map
ICPN	Interpolative Counter Propagation Network
IDFT	Inverse Discrete Fourier Transform

IFFT	Inverse Fast Fourier Transform
IIR	Infinite Impulse Response
LBG	Linear Basis Function
LMS	Least Means Square
LPC	Linear Predictive Coding
LTM	Long Term Memory
MEM	Maximum Entropy Method
MLM	Maximum Likelihood Method
MPS	Modulating Parameter Structure
MSE	Mean Square Error
MUSIC	Algorithm for Multiple Signal Classification Method
NIE	Network Input Element
NOE	Network Output Element
OCM	Output Connection Map
PAA	Parameter Adaptation Automata
PCS	Parameter Control State
PF	Projective Field
PK	Processing Kernel
PSD	Power Spectral Density
PSTF	Parameter State Transition Function
RBF	Radial Basis Function
RF	Receptive Field
RFBS	Received Feedback Signal
RLS	Recursive Least Square
RMS	Root Mean Square
SNR	Signal to Noise Ratio
SOM	Self Organising Map
STF	State Transition Function
STFT	Short Time Fourier Transform
STM	Short-Term Memory
TFBS	Transmitting Feedback Signals
TFBV	Transmitting Feedback Vector

UDN	Uni-Directional Neuron
VLSI	Very Large Scale Integrated circuit

Appendix B

Mackey-Glass Chaotic Time Series

Chaotic time series is a deterministic and non-linear series. Mackey-Glass chaotic time series is generated from the following delay equation :

$$\frac{dx(t)}{dt} = \frac{ax(t-\tau)}{1+x^c(t-\tau)} - bx(t \qquad \text{(B.1)}$$

In this work the following constants are used :

$$a = 0.2 \;;\; b = 0.1 \;;\; c = 10 \qquad \text{(B.2)}$$

Therefore the Eq. B.1 is rewritten as :

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \qquad \text{(B.3)}$$

where the τ value determines the chaotic behaviour of the function. Choosing the τ value > 17 produces a chaotic behaviour. In this experiments, τ equal to 30 is used. This value is similar to the model that has been used by Wang and Mendel (Wang and Mendel, 1992).

To generate the time series, Euler method is used to solve the Eq. B.3. The following initial values are used to generate the time series:

$x(0) = 0.7$

$x(t) = x(0) + 0.02 \,.\, t \quad ; t = 1, 2, \ldots, 30$

After t > 30, the Euler method is applied to produce x(t) where t = 31, ..., 1000. This time series is shown in Figure B.1.

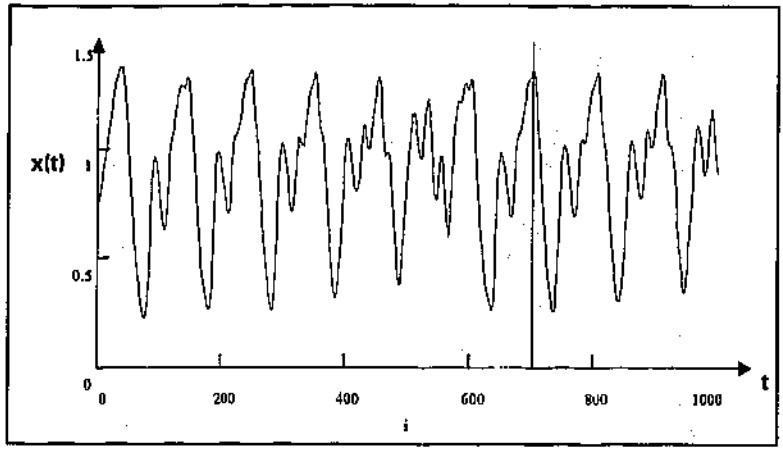

Figure B.1 Mackey-Glass chaotic time series

Appendix C

Example of Real Background Noise Elimination

In this simulation 3 kinds of real noise are used. The noises are taken from CD sample (Spectacular Sound Effect, EMI Record Ltd, 1990). Those noises are

- Sound of Typical Street Noise.
- Sound of Crowd Cheering in a pub.
- Sound of troop marching.

Those sounds are sampled by using DSP board TMS320C25 from Dalanco Spry with sample rate is equal to 5 KHz. As the testing signal a 300 Hz sinusoidal wave which is created by signal generator is used.

The Fuzzy-CPN is trained only using one type of noise. After that the test signal is added with the same noise. This corrupted signal is fed into the system. The Fuzzy-CPN eliminates the noise from the corrupted signal and produce a noise-free signal. These steps are performed to all noise.

All these simulation have not been performed in real time. The spectrogram and the spectral subtraction and FFT is performed by using separate software. The Fuzzy-CPN simulator software are only supplied by the spectrogram and produces the average periodogram.

Firstly, the noises are represented by using conventional spectrogram method and the proposed spectrogram method in order to compare the both methods. It shows that the proposed method provide a clearer representation in order to detect the noise. Secondly the simulation of background noise elimination using those noises are shown.

Street Noise

This noise is typical noise of a street, there is sound of car, people walking, people talking. This noise is combination of the stationary noise from the street background and non-stationary noise, such as car, horn etc..

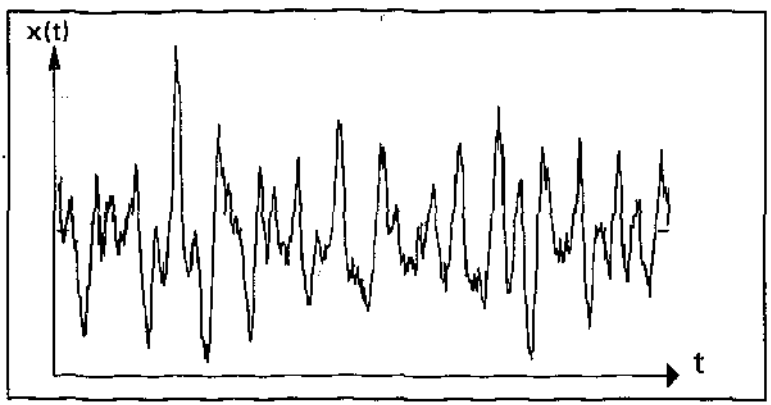

Figure C.1 Time-domain representation of street noise

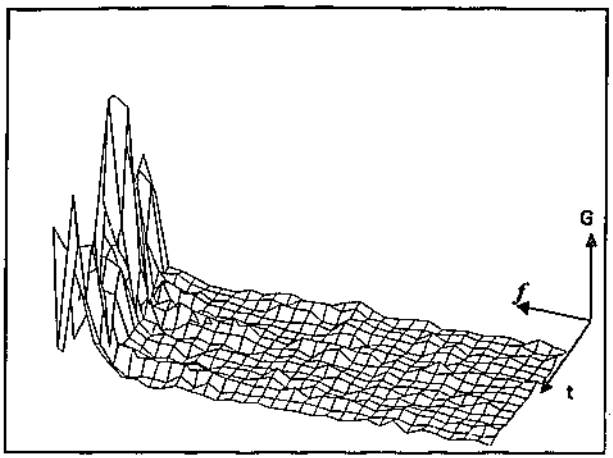

Figure C.2 Conventional spectrogram of signal shown in Figure C.1

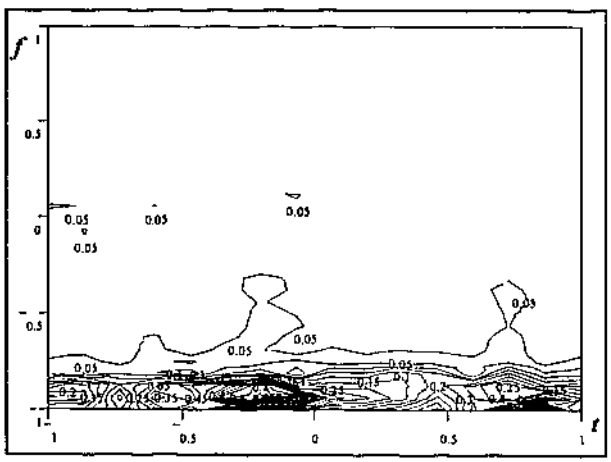

Figure C.3 Contour diagram of Figure C.2

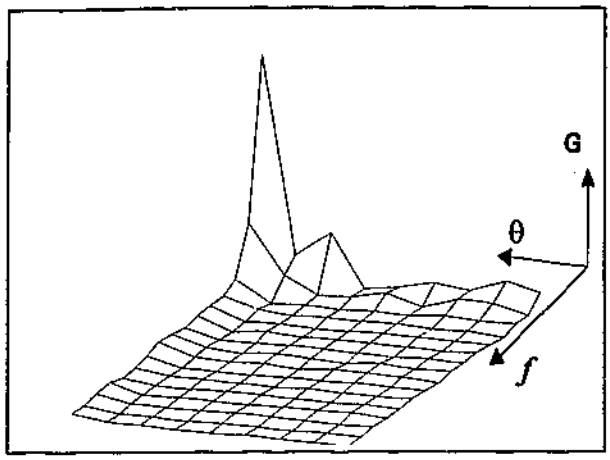

Figure C.4 Proposed spectrogram of signal shown in Figure C.1

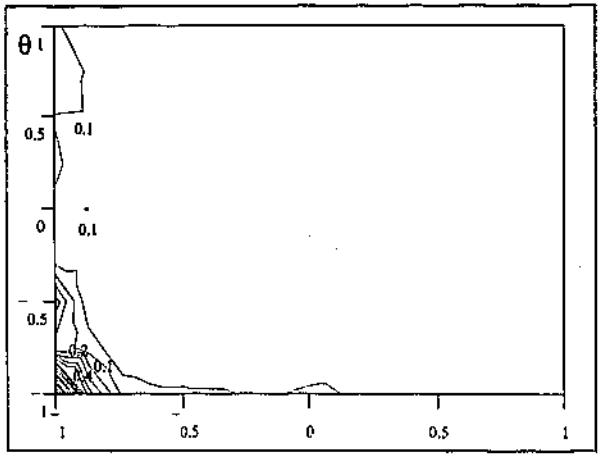

Figure C.5 Contour diagram of Figure C.4

Foot Step noise

This transient noise is recorded from the sound of foot step in the silence environment. This noise is an example of non-broadband noise and non-gaussian noise.

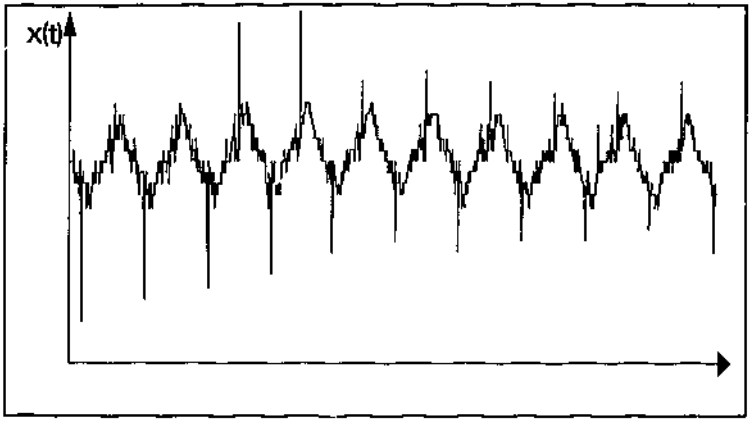

Figure C.6 Time-domain representation of foot step noise

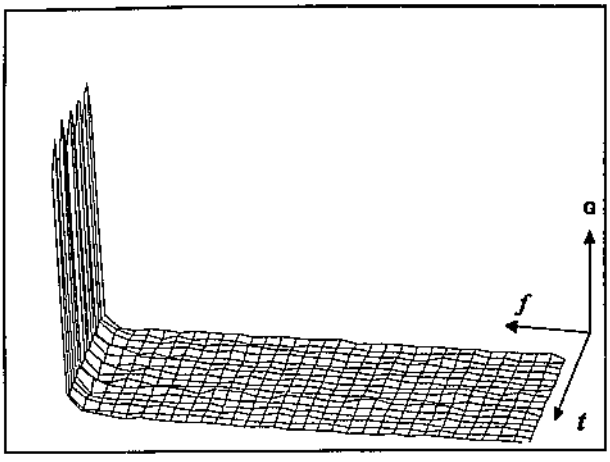

Figure C.7 Conventional spectrogram of signal shown in Figure C.6

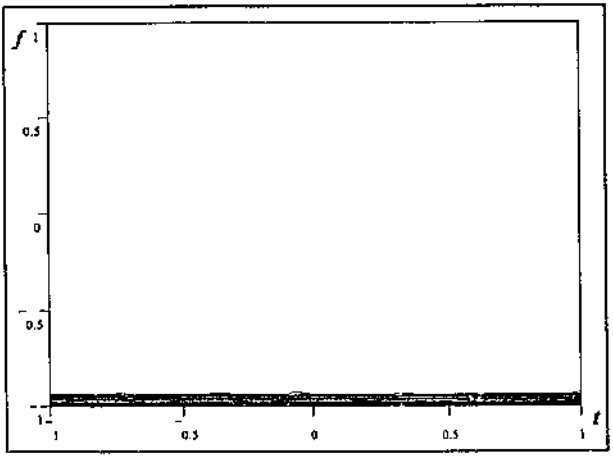

Figure C.8 Contour diagram of Figure C.7

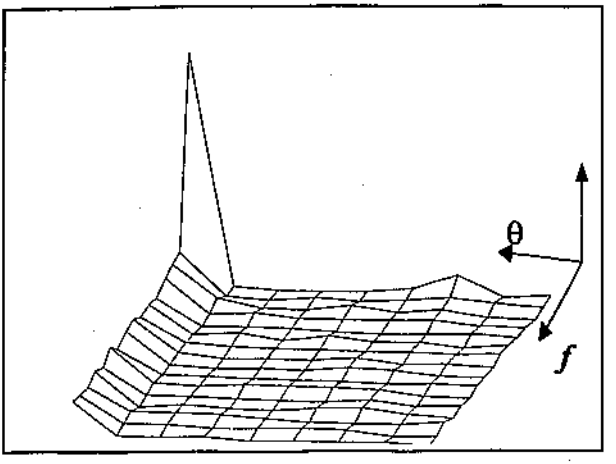

Figure C.9 Proposed spectrogram of signal shown in Figure C.6

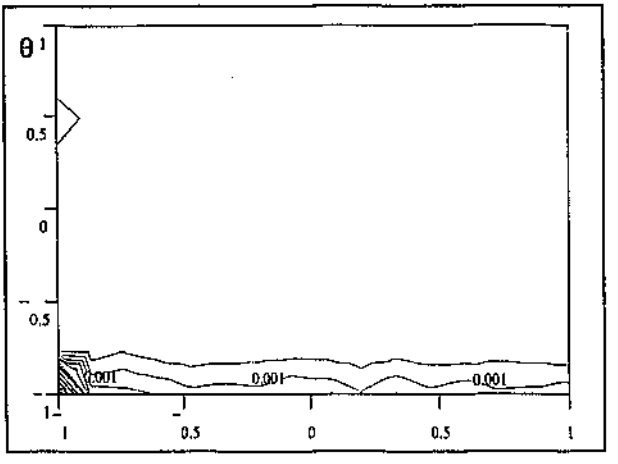

Figure C.10 Contour diagram of Figure C.9

Crowded (people talking in a pub) noise

This noise is example of broadband noise and stationary noise. However, this noise is taken from the real environment which is non-gaussian.

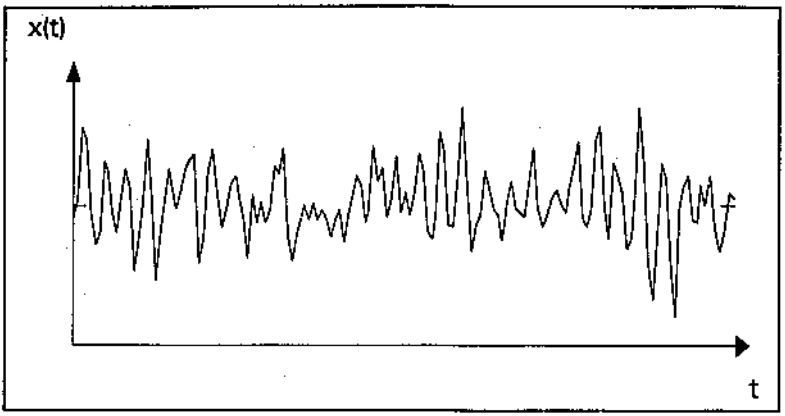

Figure C.11 Time-domain representation of crowded noise

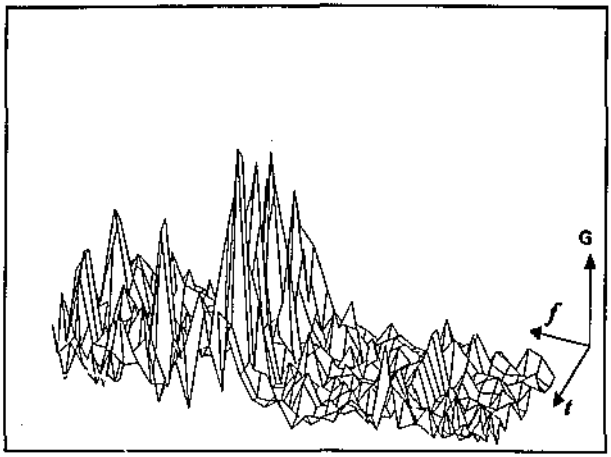

Figure C.12 Conventional spectrogram of signal shown in Figure C.11

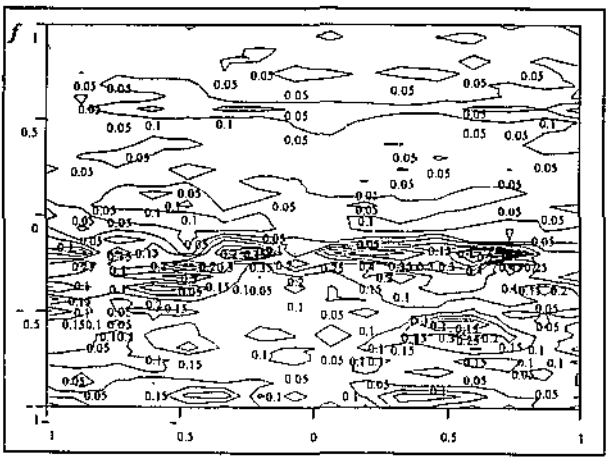

Figure C.13 Contour diagram of Figure C.12

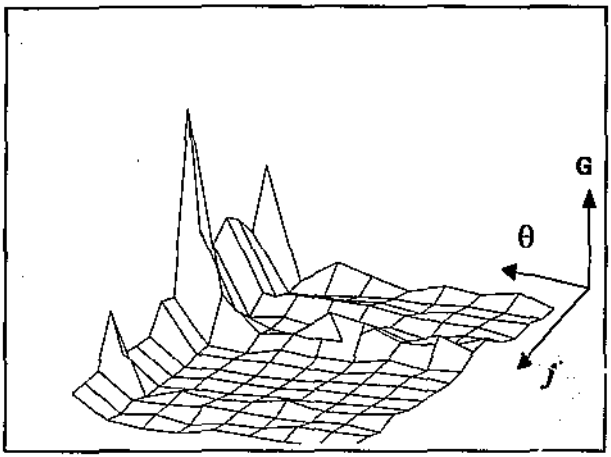

Figure C.14 Proposed spectrogram of signal shown in Figure C.11

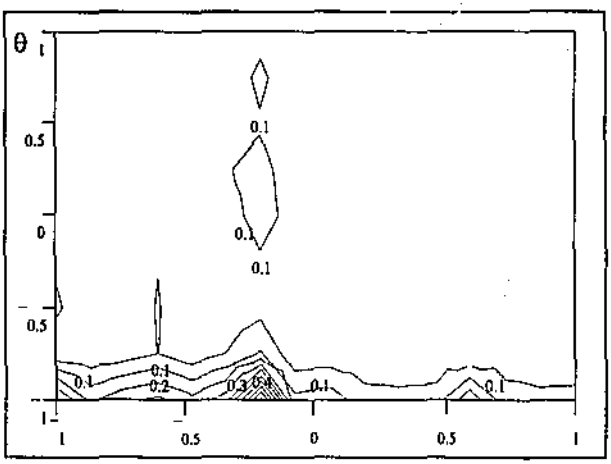

Figure C.15 Contour diagram of Figure C.14

Result of Real Noise Filtering

Street Noise

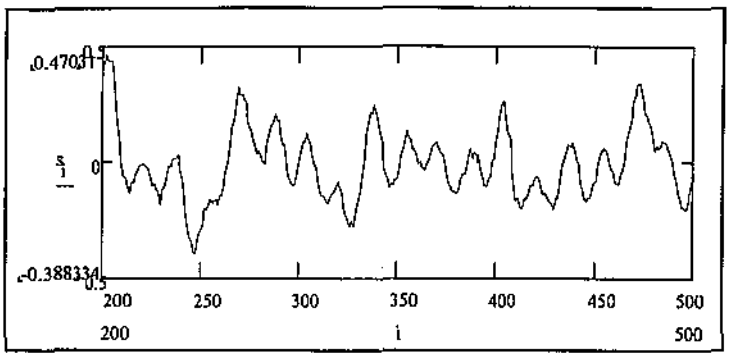

Figure C.16 Input signal : sinousidal and street noise

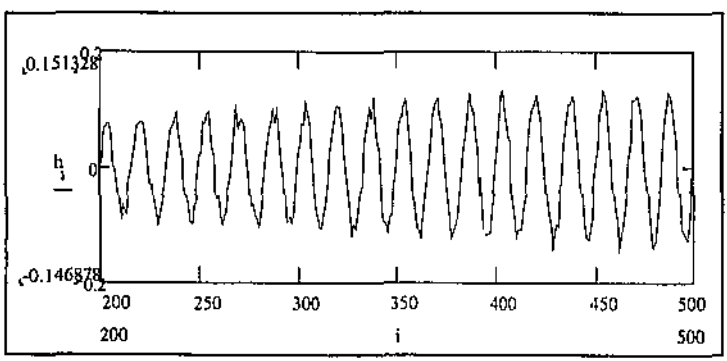

Figure C.17 Output signal

Crowd Cheering

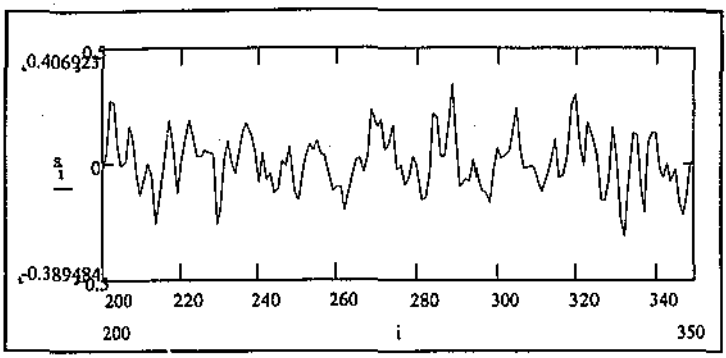

Figure C.18 Input signal : sinusoidal with crowd cheering noise

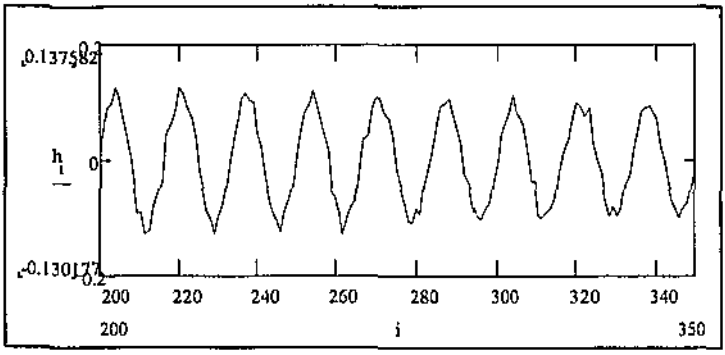

Figure C.19 Output signal

Troop Marching

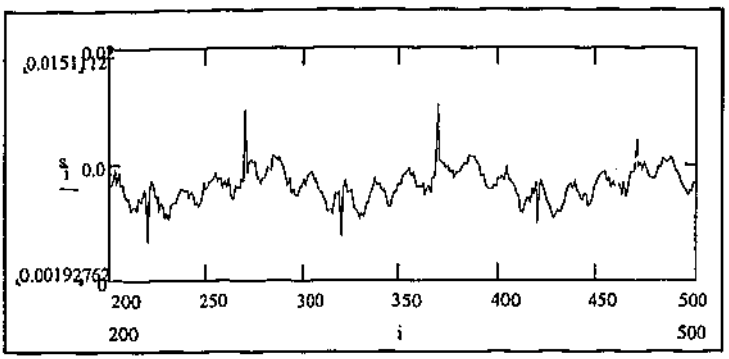

Figure C.20 Input signal : sinusoidal with troop marching noise

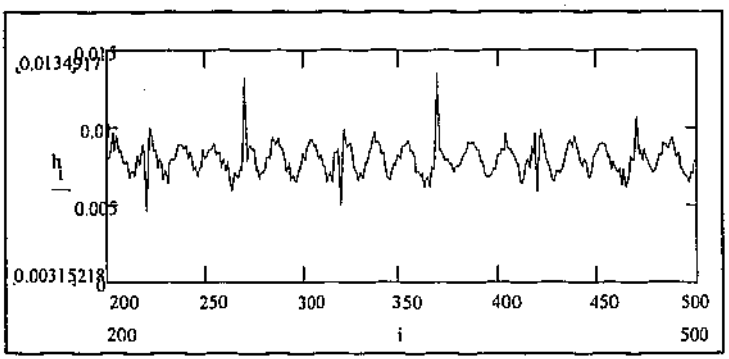

Figure C.21 Output signal

Appendix D

List of Equations

$$y(n) = \sum_{k=-M}^{M} w(k)x(n-k) \tag{2.1}$$

$$x(k) \overset{F}{\leftrightarrow} X(e^{j\omega}) \tag{2.2}$$

$$T(e^{j\omega}) = \Psi(X(e^{j\omega}), Y(e^{j\omega})) \tag{2.3}$$

$$x(k) \otimes y(k) \overset{F}{\leftrightarrow} X(e^{j\omega}) \cdot Y(e^{j\omega}) \tag{2.4}$$

$$\mathbf{x}_k = [x_k, x_{k-1}, \ldots, x_{k-L+1}]^T \tag{2.5}$$

$$\mathbf{w} = [w_0, w_1, \ldots, w_{L-1}]^T \tag{2.6}$$

$$e(k) = d(k) - y(k) \tag{2.7}$$

$$r_{dx}(n) = E[d_x x_{k+n}] \tag{2.8}$$

$$r_{xx}(n) = E[x_k x_{k+n}] \tag{2.9}$$

$$\mathbf{R} = E\left[\mathbf{x}_k \mathbf{x}_k^T\right] \tag{2.10}$$

$$R = \begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \ldots & r_{xx}(L-1) \\ r_{xx}(1) & r_{xx}(0) & \ldots & r_{xx}(L-2) \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ r_{xx}(L-1) & r_{xx}(L-2) & \ldots & r_{xx}(0) \end{bmatrix} \tag{2.11}$$

$$\mathbf{P} = E[d_x \mathbf{x}_x] = E[\mathbf{x}_k d_k] \tag{2.12}$$

$$MSE = E\left[e_k^2\right] = E\left[(d_k - y_k)^2\right]$$
$$= E\left[d_k^2\right] + E\left[y_k^2\right] - 2E[d_k y_k]$$
$$= r_{dd}(0) + r_{yy}(0) - 2r_{dy}(0) \tag{2.13}$$

$$MSE = r_{dd}(0) + \sum_{i=0}^{L-1}\sum_{l=0}^{L-1} w_i w_m r_{xx}(i-m) - 2\sum_{i=0}^{L-1} w_i r_{xd}(i) \tag{2.14}$$

$$MSE = r_{dd}(0) + \mathbf{w}^T\mathbf{R}\mathbf{w} - 2\mathbf{p}^T\mathbf{w} \tag{2.15}$$

$$\nabla = \frac{\partial(MSE)}{\partial \mathbf{w}} = \left[\frac{\partial(MSE)}{\partial w_0} \quad \frac{\partial(MSE)}{\partial w_1} \quad \cdots \quad \frac{\partial(MSE)}{\partial w_{L-1}}\right]^T$$
$$= 2\mathbf{R}\mathbf{w} - 2\mathbf{p} \tag{2.16}$$

$$\mathbf{w}^O = \mathbf{R}^{-1}\mathbf{p} \tag{2.17}$$

$$\tfrac{1}{2}\mathbf{R}^{-1}\nabla = \mathbf{w} - \mathbf{R}^{-1}\mathbf{p}$$

$$\mathbf{w}^O = \mathbf{w} - \tfrac{1}{2}\mathbf{R}^{-1}\nabla \tag{2.18}$$

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu\mathbf{R}^{-1}\nabla_k \tag{2.19}$$

$$\mathbf{R}_k^{-1} = \delta\, \mathbf{I}_N \tag{2.20}$$

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu\hat{\nabla}_k \tag{2.21}$$

$$\Delta J = J - J_{\min} \tag{2.22}$$

$\Delta J > 0$ for $\mathbf{v} \neq 0$, where $\mathbf{v} = \mathbf{w}^O - \mathbf{w}$

$$\nabla_k = \frac{\partial(J)}{\partial \mathbf{w}_k} \tag{2.23}$$

$$\hat{\nabla}_k = \frac{\partial[e^2(k)]}{\partial \mathbf{w}_k} = 2e_k\frac{\partial e_k}{\partial \mathbf{w}_k}$$
$$e_k = d_k - \mathbf{w}_k^T\mathbf{x}_k$$
$$\hat{\nabla}_k = 2e(k)\frac{\partial}{\partial \mathbf{w}_k}\{d(k) - \mathbf{w}^T\mathbf{x}(k)\} = -2e(k)\frac{\partial}{\partial \mathbf{w}_k}\{\mathbf{w}^T\mathbf{x}(k)\}$$
$$= -2e(k)\mathbf{x}(k) \tag{2.24}$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e(k)\mathbf{x}(k) \tag{2.25}$$

$$y(k) = \mathbf{w}^T(k)\,\mathbf{x}(k) \tag{2.26}$$

$$e(k) = d(k) - y(k) \tag{2.27}$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\, e(k)\,\mathbf{x}(k) \tag{2.28}$$

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mu\hat{\mathbf{R}}_k^{-1}\hat{\nabla}_k \tag{2.29}$$

$$\mathbf{R}_{k+1} = \mathbf{R}_k + \mathbf{x}(k)\,\mathbf{x}^T(k) \tag{2.30}$$

$$\mathbf{p}_{k+1} = \mathbf{p}_k + d(k)\,\mathbf{x}(k) \tag{2.31}$$

$$\mathbf{R} \rightarrow \mathbf{R}^{-1} \tag{2.32}$$

$$\mathbf{w}_{k+1}^{O} = \mathbf{R}_{k+1}^{-1}\mathbf{p}_{k+1} \tag{2.33}$$

$$\mathbf{R}_{k+1}^{-1} = \mathbf{R}_k^{-1} - \frac{\mathbf{R}_k^{-1}\mathbf{x}(k)\mathbf{x}(k)\mathbf{R}_k^{-1}}{1 + \mathbf{x}^T(k)\mathbf{R}_k^{-1}\mathbf{x}(k)} \tag{2.34}$$

$$\mathbf{w}_{k+1}^{\mathbf{O}} = \mathbf{w}_k^{\mathbf{O}} + \frac{e(k)\cdot\mathbf{z}_k}{1+q} \tag{2.35}$$

$$\mathbf{z}_k \stackrel{\Delta}{=} \mathbf{R}_k^{-1}\mathbf{x}(k) \tag{2.36}$$

$$q = \mathbf{x}^T(k)\mathbf{z}_k \tag{2.37}$$

$$v = \frac{1}{1+q} \tag{2.38}$$

$$\mathbf{z}_k = v\cdot\mathbf{z}_k \tag{2.39}$$

$$\mathbf{w}_{k+1}^{o} = \mathbf{w}_k^{o} + e_0(k)\tilde{\mathbf{z}}_k \tag{2.40}$$

$$\mathbf{R}_{k+1}^{-1} = \mathbf{R}_k^{-1} - \tilde{\mathbf{z}}_k\mathbf{z}_k^T \tag{2.41}$$

$$C_{RLS} = (L-N+1)\cdot 2N^2 + (L-N+1)\cdot 4N \tag{2.42}$$

$$N = \{F, q, \mathbf{w}, \Gamma, C\} \tag{3.1}$$

$$F = \{\beta, \Phi, \theta\} \tag{3.2}$$

$$y = \Phi(\beta(\mathbf{x}|\mathbf{w}), \theta) \tag{3.3}$$

$$\beta(\mathbf{x}|\mathbf{w}) = \sum_{j=1}^{n} w_{ij} x_j \tag{3.4}$$

$$\beta(\mathbf{x}|\mathbf{w}) = \sqrt{\sum_{j=1}^{n} (x_j - w_{ij})^2} \tag{3.5}$$

$$\Phi(u) = \frac{1}{1 + e^{-\frac{u}{\sigma}}} \tag{3.6}$$

$$\Phi(u) = ce^{-\frac{u^2}{\sigma^2}} \tag{3.7}$$

$$y = \Phi\left(\sum_{j=1}^{n} w_j u_j - \theta\right) \tag{3.8}$$

$$y = \Phi\left(\sqrt{\sum_{j=2}^{n} (w_{ij} - x_j)^2} - \theta\right) \tag{3.9}$$

$$\Gamma = \{\delta, \sigma\} \tag{3.10}$$

$$\mathbf{w}^{t+1} = \delta(\vec{x}^t, q^t, \mathbf{w}^t, e^t_{in}, C^t) \tag{3.11}$$

$$C^{t+1} = \sigma(\vec{x}^t, q^t, \mathbf{w}^t, e^t_{in}, C^t) \tag{3.12}$$

$$G = \{A_1, A_2, ..., A_n\} \tag{3.13}$$

$$A_i = [N_i, R_i, T_i] \tag{3.14}$$

$$\dot{\mathbf{G}} \neq 0 \tag{3.15}$$

$$\rho(H(\mathbf{x}|\mathbf{w}^O), f(\mathbf{x})) \leq \rho(H(\mathbf{x}|\mathbf{w}), f(\mathbf{x})) \tag{3.16}$$

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \varepsilon^t L(\mathbf{x}^t, \mathbf{w}^t) \tag{3.17}$$

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \varepsilon^t L(\mathbf{x}^t, \mathbf{y}^t, \mathbf{w}^t) \tag{3.18}$$

$$C(w) = \int_X J(x,w)p(x)dx = E_x\{J(x,w)\} \tag{3.19}$$

$$\nabla C(w) = E_x\{\nabla_w J(x,w)\} = 0 \tag{3.20}$$

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \gamma^t \nabla_w J(\mathbf{x}^t, \mathbf{w}^t) \tag{3.21}$$

$$A = [x, m_A(x)], \tag{3.22}$$

$$m_A(x) = \text{Degree}(x \in A) \tag{3.23}$$

$$\mu_{A_i^j}(x_i) = a_i^j \exp\left(-\frac{1}{2}\left(\frac{x_i - \bar{x}_i^j}{\sigma_i^j}\right)^2\right) \tag{3.24}$$

$$\mu_{A_i^j}(x_i) = a_i^j\left(1 - \frac{\left|x_i - \bar{x}_i^j\right|}{\sigma_i^j}\right) \tag{3.25}$$

$$\mu_{A_i^j}(x_i) = f\left(x_i \middle| \bar{x}_i^j, \sigma_i^j\right) \tag{3.26}$$

$$R_j: \text{IF } x_1 \text{ is } A_1^j \text{ and } x_2 \text{ is } A_2^j \text{ and ...and } x_n \text{ is } A_n^j \text{ THEN } z \text{ is } B^j \tag{3.27}$$

$$\bar{y} = \frac{\int_{-\infty}^{\infty} y\,\mu(y)\,dy}{\int_{-\infty}^{\infty} \mu(y)dy} \tag{3.28}$$

$$f(x) = \frac{\sum_{j=1}^{K}\left(\bar{z}^j\,\mu^j\right)}{\sum_{j=1}^{K}\left(\mu^j\right)} \tag{3.29}$$

$$S: I^{n_1} \times \ldots \times I^{n_r} \to I^{p_1} \times \ldots \times I^{p_s} \tag{3.30}$$

$$S: I^n \to I^p \tag{3.31}$$

$$\mu = \Omega(d|\rho) \tag{4.1}$$

$$d = \Psi(\mathbf{x}|\mathbf{w}) \tag{4.2}$$

$$\mu(d) = \left(1 + \left(\frac{d}{(r-d)}\right)\right)^{-1} \tag{4.3}$$

$$\Psi(\mathbf{x}) = \sqrt{\sum_{j=1}^{n} (x_j - w_j)^2} \tag{4.4}$$

$$y = G \cdot M(\mathbf{u}|\mathbf{w}) \tag{4.5}$$

$$M = \sum_{j=1}^{p} u_j w_j \tag{4.6}$$

$$G = \frac{1}{\sum_{j=1}^{p} u_j} \tag{4.7}$$

$$\phi : R^n \rightarrow R^m, \text{ where } y_i = \phi(x_i) \tag{4.8}$$

$$G_{Fuzzy\text{-}CPN} = \{I_1, .., I_n, M_1, ..., M_l, O_1, ..., O_k, C\} \tag{4.9}$$

$$I = [IP, R_{IP\text{-}I}, T_{I\text{-}M}] \tag{4.10}$$

$$M = [FN, R_{I\text{-}M}, \{T_{M\text{-}O}, T_{M\text{-}C}\}] \tag{4.11}$$

$$O = [DFN(R_{O^G\text{-}C}), R_{O\text{-}M}, T_{O\text{-}OP}] \tag{4.12}$$

$$C = [DCN, R_{M\text{-}C}, T_{C\text{-}O^G}] \tag{4.13}$$

$$d_i = \sqrt{\sum_{j=1}^{n} (c_{ji} - X_j)^2} \tag{4.14}$$

$$\mu_i = \left(1 + \left(\frac{d_i}{(r_i - d_i)}\right)\right)^{-1} \tag{4.15}$$

$$W1_{win}^{new} = W1_{win}^{old} + \beta\left(X - W1_{win}^{old}\right)$$

$$W2_{win}^{new} = W2_{win}^{old} + \beta\left(Y - W2_{win}^{old}\right) \tag{4.16}$$

$$\beta = \frac{\alpha}{\alpha + 1} \tag{4.17}$$

$$\alpha_{win}^{new} = \left(\frac{1}{\alpha_{win}^{old}} + 1\right)^{-1} \tag{4.18}$$

$$r_{win}^{new} = r_{win}^{old} + \sqrt{\sum_{j=1}^{n}\left(W1_{j,win}^{new} - W1_{j,win}^{old}\right)^2} \tag{4.19}$$

$$W1^{new} = X; \quad W2^{new} = Y; \; r^{new} = \rho \; ; \qquad \alpha^{new} = 1 \tag{4.20}$$

$$W1^{ann} = X \; ; \quad W2^{ann} = Y \quad ; \quad r^{ann} = \rho \quad ; \; \alpha = 1 \tag{4.21}$$

$$S_i = \left(\sqrt{\sum_{j=1}^{n}\left(W1_{j,i} - X_j\right)^2}\right) - r_i \tag{4.22}$$

$$Out_{DCN} = \sum_{i=1}^{p} \mu_i \tag{4.23}$$

$$Out_k = G\left(\sum_{i=1}^{p}\left(\mu_i \, W2_{ik}\right)\right) \tag{4.24}$$

$$G = \frac{1}{Out_{DCN}} \tag{4.25}$$

$$Out_k = \frac{\sum_{i=1}^{p}\left(\mu_i W2_{ik}\right)}{\sum_{i=1}^{p}\mu_i} \tag{4.26}$$

$$f: (x(t), x(t-1), \ldots, x(t-k), S) \rightarrow y(t) \tag{4.27}$$

$$g: (x(t), x(t-1), \ldots, x(t-k), S) \rightarrow \hat{y}(t) \tag{4.28}$$

$$f(x)=\begin{cases} x^3-x^2+0.03 \text{ ; for } x<0.5 \\ x^3-x^2 \qquad\quad \text{ ; for } x\geq 0.5 \end{cases} \tag{5.1}$$

$$MSE=\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i-y_i)^2 \tag{5.2}$$

$$y=\frac{m_1y_1+m_2y_2+m_3y_3}{m_1+m_2+m_3} \tag{5.3}$$

$$m_n=\left(\sqrt{\sum(x_i-c_i^n)^2}\right)^{-1} \tag{5.4}$$

$$y=4x(1-x) \tag{5.5}$$

$$f:[x(k),x(k-1),\ldots x(k-p)]\rightarrow x(k+q) \tag{5.6}$$

$$\frac{dx(t)}{dt}=\frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)}-0.1x(t) \tag{5.7}$$

$$f:R^9\rightarrow R \tag{5.8}$$

$$f:[x(k),x(k-1),\ldots,x(k-8)]\rightarrow x(k+1) \tag{5.9}$$

$$d(k) = s(k) + \alpha(u(k)) + \omega(k) \tag{6.1}$$

$$u(k) = c(k) + m(k) \tag{6.2}$$

$$\hat{y}(k) = \beta(u(k) + \omega(k)) \tag{6.3}$$

$$e(k) = d(k) - \hat{y}(k) \tag{6.4}$$

$$e(k) = s(k) \tag{6.5}$$

$$\rho(t,f) = \int\int\int e^{j2\pi v(u-t)} g(v,t) s^*(u - \tfrac{1}{2}\tau) s(u + \tfrac{1}{2}\tau) e^{-j2\pi f\tau} dv du d\tau \tag{6.6}$$

$$g(v,\tau) = \int h^*(u - \tfrac{1}{2}\tau) e^{-j2\pi vu} h(u + \tfrac{1}{2}\tau) du \tag{6.7}$$

$$\rho(t,f) = \left| \int e^{-j2\pi f\tau} s(\tau) h(\tau - t) d\tau \right|^2 \tag{6.8}$$

$$\zeta(\theta,f) = \int |\rho(t,f)|^2 e^{j2\pi\theta t} dt \tag{6.9}$$

$$s(t) = \begin{cases} A_1 \sin(\omega_1 t) & \text{for } t < p \\ A_1 \sin(\omega_1 t) + A_2 \sin(\omega_2 t) & \text{for } p \le t < l \\ A_2 \sin(\omega_2 t) & \text{for } l \le t \end{cases} \tag{6.10}$$

$$x(k) = s(k) + n(k) \tag{6.11}$$

$$P_x(\omega) = P_s(\omega) + P_n(\omega) \tag{6.12}$$

$$P_x^l(\omega) = P_s^l(\omega) + P_n^l(\omega) \tag{6.13}$$

$$\hat{P}_s(\omega) = P_x(\omega) - \hat{P}_n(\omega) \tag{6.14}$$

$$\hat{\Gamma}_s(\omega; m) = \Gamma_x(\omega; m) - \hat{\Gamma}_n(\omega; m) \tag{6.15}$$

$$\left| \hat{S}_s(\omega; m) \right|^2 = |S_x(\omega; m)|^2 - \left| \hat{S}_n(\omega; m) \right|^2 \tag{6.16}$$

$$x_n^N = \begin{cases} x_n & \text{for n = 0, 1, 2, . . N - 1} \\ 0 & \textit{otherwise} \end{cases} \tag{6.17}$$

$$S_N(\omega) = \frac{1}{N} X_N(\omega) X_N^*(\omega) = \frac{1}{N} |X_N|^2 \tag{6.18}$$

$$X_N(\omega) = \sum_{-\omega}^{\omega} x_n^N \, e^{-jn\omega T_s} = \sum_{n=0}^{N-1} x_n \, e^{-jn\omega T_s} \tag{6.19}$$

$$X^{(i)}(n) = x(iM + n) \qquad \begin{matrix} n = 0, 1, \ldots, M - 1 \\ i = 0, 1, \ldots, K - 1 \end{matrix} \tag{6.20}$$

$$\hat{I}_M^{(i)}(\omega) = \frac{1}{M} \left| \sum_{n=0}^{M-1} x^{(i)} \, e^{-jn\omega T} \right|^2 \tag{6.21}$$

$$S_M^K(\omega) = \frac{1}{K} \left| \sum_{i=0}^{K-1} \hat{I}_M^{(i)}(\omega) \right| \tag{6.22}$$

$$SNR = 10 \log \left(\frac{P_{signal}}{P_{noise}} \right) \tag{6.23}$$

Appendix E

List of Papers

ANZIIS-93

Australian and New Zealand conference on Intelligent Information Systems 1993

Perth, Western Australia
December 1-3, 1993

Organising Committee

General Chairman: Yianni Attikiouzel, The University of Western Australia

Technical Programme Chair: Marimuthu Palaniswami, The University of Melbourne

Treasurer: Dorota Kieronska, Curtin University of Technology

New Zealand Liaison Chair: George Coghill, University of Auckland

Local Committee Chair: Keith Godfrey, The University of Western Australia

International Panel of Reviewers

K. Aihara	Tokyo Denki University, Japan
J. Andreae	University of Canterbury, New Zealand
S. Bang	Pohang Institute of Science & Technology, Korea
A. Bowles	BHP Research Laboratories, Australia
T. Caelli	The University of Melbourne, Australia
L. Cahill	La Trobe University, Australia
R. Colomb	University of Queensland, Australia
A. Constantinides	Imperial College, U.K.
J. Cybulski	La Trobe University, Australia
T. Dillon	La Trobe University, Australia
T. Downs	University of Queensland, Australia
R. Evans	The University of Melbourne, Australia
N. Foo	University of Sydney, Australia
T. Fukuda	Nagoya University, Japan
P. Hingston	The University of Western Australia, Australia
R. Hodgson	Massey University, New Zealand
A. Horsfall	Fujitsu Australia Limited, Australia
Y. Hsu	National Taiwan University, Taiwan
R. Jarvis	Monash University, Australia
A. Jennings	Telecom Research Laboratories, Australia
J. Kacprzyk	Polish Academy of Sciences, Poland
S. Kollias	National Technical University of Athens, Greece
B. Kosko	University of Southern California, U.S.A.
A. Kowalczyk	Telecom Research Laboratories, Australia
J. Morris	University of Tasmania, Australia
D. Nandagopal	D.S.T.O., Australia
T. Nguyen	University of Tasmania, Australia
L. Patnaik	Indian Institute of Science, India
A. Ramer	University of New South Wales, Australia
R. Rist	University of Technology, Sydney, Australia
T. Saito	Hosei University, Tokyo, Japan
J. Sitte	Queensland University of Technology, Australia
Z. Sha	Chinese Research Institute of Radio Prop.
R. Stone	University of Salford, U.K.
R. Uthurusami	General Motors Research, U.S.A.
A. Venetsanopoulos	University of Toronto, Canada
K. Wong	The University of Western Australia, Australia
T. Yamakawa	Kyushu Institute of Technology, Japan
A. Zomaya	The University of Western Australia, Australia

Neural Network-Based Adaptive Filtering For Background Noise Elimination

H. N. Cheung
Dept. of Computer and Communication Engineering

I Made Wiryana and J. Millar
Dept. of Computer Science

Edith Cowan University
Joondalup, Western Australia

ABSTRACT:

The elimination of background noise in applications where an uncorrupted input signal is required is not a trivial task, especially when the noise is non-deterministic and non-stationary. In addition, the duration of the noise may be short compared with the observation intervals for the input signal. This paper reports the development of a background noise elimination system which makes use of time-frequency representation of the input signal and a neural network-based adaptive filter. Modifications are made to the time-frequency transformation process to obtain better frequency representation of the signal and also to a Counter Propagation Network to have better clustering properties. Software simulation results with some typical signals corrupted by Gaussian noise are also presented.

1. Introduction

In many areas of science and engineering in which input signals are obtained using signal acquisition equipment, such as electromagnetic and acoustic surveillance (Anderson et. al., 1991, Casellman et al., 1991), seismic signal processing (Magotra et. al., 1991), speech processing (Kobatake ,1990), biomedical signal processing (Uncini, 1991), there is always a problem in detecting the presence of non-stationary random signals or noise in the background. The elimination of such background noise is not a trivial task due to its non-stationary and non-deterministic characteristics. In addition, the duration of this noise may be short compared to the observation intervals for the input signals

Various methods have been proposed to tackle this problem including fixed filtering (Boll, 1979) and adaptive filtering techniques using conventional filters. In the case of fixed filtering techniques, the unavailability of an ideal signal model and the use of predefined parameters of the filter makes this problem difficult to solve (Classen, 1983). The use of adaptive methods has improved the performance of background noise elimination to a large extent (Vary, 1983). With these methods, the parameters of the filters change to adapt to the non-deterministic characteristics of the background noise. However, provision has to be made for the filters to readapt to the change in the environment. Sometimes it may lead to the unsatisfactory performance of the overall system (Connel, 1990).

Artificial Neural Networks (ANN) offer an alternative technique for adaptive filtering. A general ANN model for adaptive filtering has been proposed by Nerrand (Nerrand et al. 1993). It has also been shown that ANN can be applied to separate a signal into different signals (Cohen et al., 1991) and to perform signal classification (Malfkoff, 1992). In the area of background noise elimination, Xue (Xue, 1992) has applied ANN based adaptive matched filter biomedical processing. This paper reports on the work on the development of an ANN model for the elimination of background noise with application to speech processing.

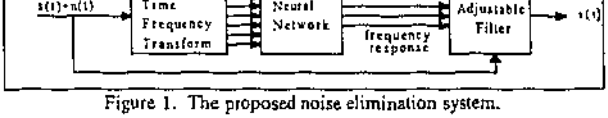

Figure 1. The proposed noise elimination system.

Shown in Fig. 1 is a block diagram of the proposed noise elimination system. In this model, the sampled input signal, corrupted by noise, is transformed into frequency domain to extract the frequency feature. Both time and frequency representations of the signal are then fed into the neural network which has been trained to reject the noise and to produce at its output the filtered input signal.

2. Time-frequency representation of input signals

Fig. 2 shows a general model for noise elimination where at the input, the sequence s[k] represents the desired input signal and u[k] any external noise signal. Inside the model, $\omega[k]$ represents internal broadband measurement noise. The problem becomes one of accurately modeling the corrupted channel $\alpha(.)$ and the internal

noise by means of the filter β(.). In general, α(.) represents a nonlinear medium. Therefore, β(.) must be a nonlinear function.

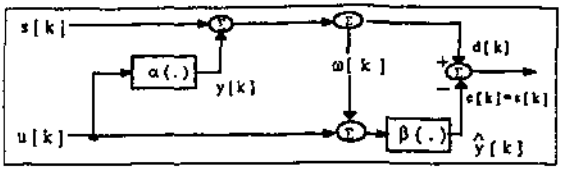

Figure 2. General model of noise elimination

The background noise u(k) can be considered as consisting of a stationary continuous noise c(k) and isolated noise m(k). In speech processing, the noise elimination process is to detect and eliminate, from the non-stationary speech source, the three noise components, i.e. the deterministic and stationary internal noise ω[k], the non-deterministic and stationary noise s(k), and more importantly the non-deterministic and non-stationary noise m(k).

Detection and classification of the signal components in time or frequency domain can be performed by firstly sliding the signal through an observation window. Normally, to extract the signal feature from either the time or frequency domain representation of the signal is not sufficient for a complete analysis of the signal. Therefore we propose to use a combined time and frequency representation.

There are several methods to generate the time-frequency distribution of a signal (Cohen, 1992). The method adopted in our model is the spectrogram approach. Given a signal, corrupted by noise, s(t), its time-frequency distribution is expressed as:

$$\rho(t,f) = \iiint e^{j2\pi v(u-t)} g(v,t) s^*(u-\tfrac{1}{2}\tau) s(u+\tfrac{1}{2}\tau) e^{-j2\pi f\tau} dv du d\tau \qquad (1)$$

where g is the kernel function.

For the spectrogram representation, the kernel function is:

$$g(v,\tau) = \int h^*(u-\tfrac{1}{2}\tau) e^{-j2\pi v u} h(u+\tfrac{1}{2}\tau) du \qquad (2)$$

By substituting Equation (2) to (1), the time-frequency distribution can be written as:

$$\rho(t,f) = \left| \int e^{-j2\pi f\tau} s(\tau) h(\tau - t) d\tau \right|^2 \qquad (3)$$

This formulation has been used to decompose a signal into different components (Cohen and Lee, 1992) using one short-time Fourier Transform(STFT) for each non-overlapped block of signal. In our approach as shown in Fig. 3, we first use the short-time Fourier Transform to get the frequency representations for overlapped blocks of signal. Then another short-time Fourier Transform is performed at the same frequency to all the resultant frequency representations to obtain the change of the frequency of the signal. The result of this manipulation is to extract finer representation of the frequency feature of the signal. The final outcome of the transformation of the signal is a two-dimensional array as shown in Fig. 3 of a single time block of the input signal. This array is then fed into the neural network.

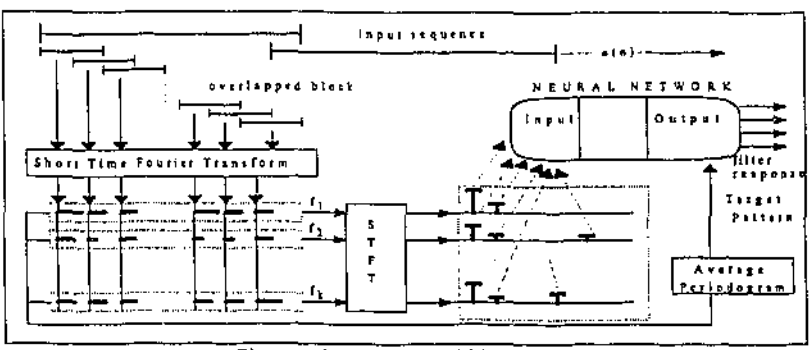

Figure 3. Input signal acquisition process.

3. Neural network and filter structures

The neural network that will be used in the model must satisfy some application constraints:

a) there is no clean speech signal for target pattern. The target pattern should be derived from the input pattern;

b) learning must be performed as fast as possible with a small number of iteration;
c) the neural network should be able to adapt its structure to minimize memory usage;
d) fast processing time is required for real-time operation.

The three-layer Counter Propagation Network (Hecht-Nielsen, 1987) is adopted in our model. This class of networks may be considered to consist of two overlapped network, i.e., the clustering network and the encoding network. The clustering network is the input and the middle layers, and the encoding network is the middle and output layers. This neural network structure is chosen for its statistical modeling and fast learning capabilities

The first layer, which is part of the clustering network, of the neural network receives the time-frequency representation of the signal. Then the clustering network performs adaptive clustering. Based on the response of the clustering network, the encoder network produces the frequency response of the filter. During training of the overall network, instead of normal input signal, noise samples are used as input to the network and the target patterns are extracted from the input patterns. The two networks will be discussed in detail in the following sections

3.1 Clustering network

Based on the standard clustering network with competitive training, we have made some modifications in the learning process to enable the overall network to avoid the stuck vector problem, to obtain better clustering, and to reorganise the middle layer in order to maximise the utilisation of the middle node.

During training, after applying the input pattern, the clustering network performs the clustering process by calculating the Euclidean distance δ_j of each template with the input pattern:

$$\delta_j = \sqrt{\sum_{i=0}^{N-1} (W1_{i,j} - I_i)^2} \qquad (4)$$

where I_i = i^{th} element of the input pattern,
N = number of nodes at input layer, and
$W1_{i,j}$ = Weight of the clustering network where the templates are stored.

After the Euclidean distances for all the templates have been calculated, the chosen cluster is the cluster with minimal value of δ. However, before the cluster is chosen, the similarity between the pattern and the input must be checked using the similarity factor ρ, to avoid the stuck vector problem in generating a neuron. If $\min(\delta) > \rho$, the cluster cannot be chosen due to its dissimilarity to the template pattern. A new cluster must be created by generating a new middle neuron. The input pattern will be stored as a new template by calculating the new values of the weights of the connections using:

$$W1_{i,free} = I_i \qquad W2_{free,k} = O_k \qquad \alpha_{free} = 1 \qquad (5)$$

where the subscript *free* is the index of the free cluster. This mechanism is similar with the ART network when the input pattern exceeds the vigilance level (Carpenter and Grossberg, 1988).

If $\min(\delta) < \rho$, the cluster may be selected and the output node of this template can be decided as the winner. The output of the clustering network will perform a 'winner-take-all' selection. The weights of the network will be updated to obtain the optimum classification of all members of the winning class. However, to obtain better clusters, we use the following formulation to update the weights so that the frequency of occurrence of a class has been selected is taken into consideration:

$$W1_{i,win}^{(t+1)} = \left(\frac{W_{i,win}^{(t)} + \alpha_{win}^{(t)} I_i}{1 + \alpha_{win}^{(t)}} \right) \qquad W2_{win,k}^{(t+1)} = \left(\frac{W2_{win,k}^{(t)} + \alpha_{win}^{(t)} O_k}{1 + \alpha_{win}^{(t)}} \right) \qquad (6)$$

where $W1_{i,j}^{(t+1)}$ = the weight value of clustering network after adjusting,
$W2_{j,k}^{(t+1)}$ = the weight value of the encoder network after adjusting,
win = the index of the node which wins the competition among the middle nodes, and
O_k = k^{th} element of the output pattern, or target pattern.

The α depends of how often this class has been chosen. This value can be calculated by using equation (7), and represents the information which is gained during the training process. If there is more information which is gained in the learning process, the α value will decrease.

$$\alpha_j^{(t+1)} = \left(1 + \frac{1}{\alpha_j^{(t)}} \right)^{-1} \qquad (7)$$

Normally, when the neurons in the middle layer are fully utilized, training can still be performed by adjusting the weights but no new neurons can be created or no new class will be created. However, in our model, when the middle layer is fully utilized and if training is required, then the network will first look for any neuron which can be reused based on the α value of each node. The neuron which has the highest α value and exceeds the threshold value can be reused as a new cluster node, thus a new class will be created using this neuron.

However, if all the neurons have α value less than the threshold, it means that all neurons have gathered enough information during training and $\min(\delta) > \rho$, which means that the input pattern is too far from all stored templates, then α' in Equation (6) will be assigned a small value and the cluster with the minimum distance will be

chosen. However the α^{m1} value for this node is not changed. In this case, the network will be attracted to the new position only with a small displacement.

3.2 Encoder network and output node connection

In this layer, the training is supervised, and the output pattern from the cluster network and the target pattern extracted from the overall input are fed into the encoding network. The output training pattern is derived from averaging values of the spectrum of the training noise. The weights will be adjusted to match the input and the output pattern using the same approach as that in the clustering network. The adjustment of the weight is made according to Equation (6).

In normal operation, the encoder networks performs a mapping of the input from the middle layer and produces at its output the response of the external input signal to the neural network at every frequency. This response is then fed into the filter and is used to update the filter coefficients. The filter equation will be generated and applied to the external input signal to produce the output signal.

The adjustable filter can be formed by using FIR/IIR filter or by using resonator bank filter. In this system we use the frequency domain filter. Input to the filter are the external input and the output from the neural network. Since the frequency domain representation of the input signal is already available from the time-frequency representation, we can take advantage of this by performing frequency domain filtering. This structure has the problem in block boundaries, and may lead to a small discontinuity in y(n). However, this problem is solved in our model by overlapping the blocks.

4. Simulation

The software simulation for the system has been performed and the results are given below. For simulation purposes, Gaussian noise with various variances are used for training the network. The size of the network used consists of 16*16 nodes for the input layer, only 20 neurons in the middle layer, and 16 in the output layer. A small number of neurons are used in the middle layer so as to test the behavior of the generation and annihilation of neurons in the clustering network.

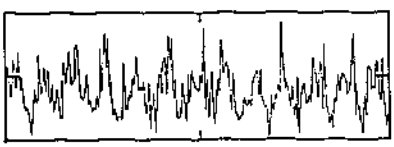

Figure 4 (a) Sinusoidal input signal is corrupted by gaussian noise with SNR equal to 17.49 dB

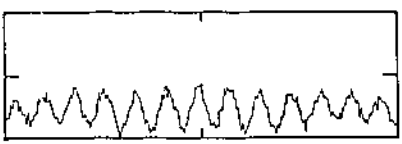

Figure 4 (b). Output signal after filtering, with SNR equal to 27.07 dB.

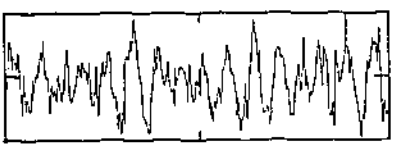

Figure 5 (a). Sinusoidal input signal with Amplitude Modulated, with SNR equal to 20.31 dB

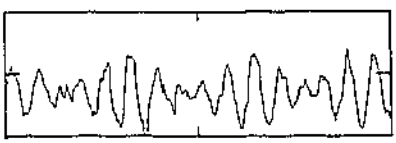

Figure 5(b). Output signal after filtering, with SNR equal to 25.52 dB.

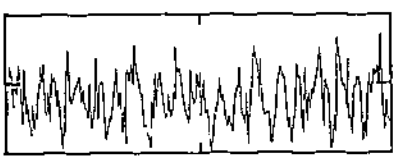

Figure 6(a). Sinusoidal input signal with frequency modulated, with SNR equal to 12.37 dB.

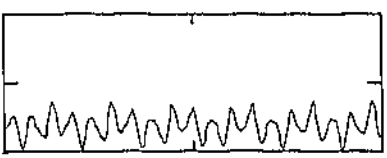

Figure 6(b). Output signal after filtering with SNR equal to 29.03 dB.

Fig. 4 (a) shows the input sinusoidal signal corrupted with Gaussian noise with SNR equal to 17.49 dB and Fig. 4 (b) the resulted output signal from the system with an improved SNR equal to 27.07 dB To test the magnitude response of the system, sinusoidal signal which has been amplitude modulated, shown in Fig. 5(a) is fed into the system. The original signal is almost recovered with SNR 25.52, with its amplitude unchanged. The frequency selectivity of the system is tested by a frequency modulated signal as shown in Fig. 6(a). The results in Fig. 6(b) shows that the original signal is recovered but with a small phase shift. The reason for the phase shift is the use of

frequency domain filtering for the filter. The problem can be solved by using resonator bank filters instead of frequency domain filter.

5. Conclusions

A noise elimination system has been developed in this paper. The system makes use of the both time and frequency representation of a signal as input to a neural network based adaptive filter. The formulation of the time frequency representation has been given and modifications of the transformation process to using overlapped blocks of signal to produce the frequency representation of the signal to overcome the boundary problem. In addition, the use of a second short-time Fourier Transform on the spectrogram to extract finer details on the frequency changes of the signal results in better representation on the frequency feature of the signal.

For the neural network in the system, modifications are also made to the Counter Propagation Network so that the problem of stuck vector is solved and the middle layer is re-organized so that with better clusters resulted. Software simulation of the system has been performed to test the response of the system to various signals corrupted by Gaussian noise, and the results of the simulation are promising.

6. Acknowledgment

The authors would like to thank the University Research and Development Committee of Edith Cowan University for their support of this research.

7. References

Anderson, James A., Michel T. Gately, P. Andrew Dens, R. Collins, (1990), Radar signal categorization using a neural network, *Proceeding of IEEE*, 78 (1), 1646-1657.

Boll, S. F., (1979), Suppression of acoustic noise in speech using spectral subtraction, *IEEE Transaction on Acoustic, Speech, an Signal*, ASSP-27, 113-120.

Carpenter, Gail A., Stephen Grossberg, (1988), The ART of adaptive pattern recognition by a self-organizing neural network, *Computer*, March, pp. 77-88.

Casselman, F. K., D. F. Freeman, D. A. Kerrigan, S. E. Lane, Millstrom, W. G. Nichols Jr., (1991), A neural network-based passive sonar detection and classification design with a low false alarm rate, *IEEE Conference on Neural Networks for Ocean Engineering*, 49-55.

Classen, T. A. C. M. and W. F. G. Mecklenbrauker (1983), Overview of adaptive techniques in signal processing, In Schussler H. W. (Ed), *Signal processing II ; Theories and Application*, Elsevier Science Publisher B. V, North-Holland, 747-754.

Cohen. L., (1992), Introduction: A Primer on Time-Frequency Analysis, In Boashash. B (Ed.), *Time Frequency Analysis, Methods and Applications*, New York ; Longmann, 17-73.

Cohen, L. Chongmoon Lee, (1992), Instantaneous Bandwidth, In Boashash. B (Ed.), *Time Frequency Analysis, Methods and Applications*, New York ; Longmann, 98-117.

Cohen, M. H., P. O. Pouliquen, A. G. Andreou, (1991), An Auto-adaptive synthetic neural network for real-time separation of independent signal sources, *IJCNN-91-Seattle ; International Joint Conference on Neural Network, I*, 211-214.

Connel, J. M., and C. S. Xydeas, (1990), A comparison of acoustic noise cancellation techniques for telephone speech, *IASTED Proceedings*

Hecht-Nielsen, Robert, (1987), Counter Propagation Network, *Proceeding of the IEEE First International Conference on the Neural Networks*.

Kobatake, Hidefumi and Kaoru Gyoutoku,(1990), Restoration of speech Contamined by Nonstationary Noises, *ISSPA 1990*.

Magotra, N., D. Hush, J. Bibbo, E. Choel, (1991), Seismic Signal Discrimination using Adaptive System Parameters, *Proceeding of the 33rd Midwest Symposium on Circuits and Systems*, 84-87.

Malkoff, Donald. B., (1990), Detection and Classification by Neural Networks and Time-Frequency Distributions, ISSPA 1990, 922-925.

Matthews, Michael B., G.S. Moschytz. (1990). *Neural network nonlinear adaptive filtering using extended Kalman filter algorithm*, Paper presented at International Neural Network Conference 1990, Paris, France., 115-118.

Nerrand, O., P. Roussel-Ragot, L. Personnaz, G. Dreyfus, (1993). Neural networks and non linear adaptive filtering: unifying concepts and new algorithms. *Neural Computation*, 5, 165-199.

Uncini, A., M . Marchesi, G. Orlandi, F. Piazza. (1990). An adaptive neural networks filters for evoked potentials. *1990 IEEE International Symposium on Circuits and Systems*, 2, 1086-1089.

Vary, Peter. (1983). On the enhancement of noisy speech. In H.W Schussler (Ed.), *SIGNAL PROCESSING II: Theories and Application.*, North-Holland:Elsevier Science Publisher B.V, 327-330.

Xue, Q. Y. Hu, W.J. Thomas, (1992), Neural Network-based Adaptive Matched Filtering for QRS Detection, IEEE Transaction on Biomedical Engineering., 39 (4), pp. 317-329.

Proceedings of the Fifth Australian Conference on Neural Networks

ACNN'94

Brisbane

31st January to
2nd February 1994

Edited by

A.C. Tsoi and T. Downs
Department of Electrical and Computer Engineering
University of Queensland
St Lucia, Qld 4072
Australia

copyright University of Queensland Electrical and Computer Engineering 1994

ACNN'94 Organizing Committee

General Chair
Prof Ah Chung Tsoi, University of Queensland

Technical Chair
Prof Tom Downs, University of Queensland

Technical Committee
Prof Yianni Attikouzel, University of Western Australia
Dr Peter Bartlett, Australian National University
Prof Robert Bogner, University of Adelaide
Prof Terry Caelli, University of Melbourne
Dr George Coghill, University of Auckland
Prof Max Coltheart, Macquarie University
Dr Phil Diamond, University of Queensland
Prof Joachim Diederich, Queensland University of Technology
Prof Tom Downs, University of Queensland
Dr Simon Goss, Defence Science & Technology Organisation
Prof Graeme Halford, University of Queensland
A/Prof Richard Heath, University of Newcastle
Dr Michael Humpreys, University of Queensland
Dr Marwan Jabri, Sydney University
A/Prof Andrew Jennings, Royal Melbourne Institute of Technology
Prof Bill Levik, Australian National University
Dr Adam Kowalczyk, Telecom
Prof Dennis Longstaff, University of Queensland
Dr D Nandagopal, Defence Science & Technology Organisation
Dr M Palaniswami, University of Melbourne
Dr Jack Pettigrew, University of Queensland
Dr Nick Redding, Defence Science & Technology Organisation
Dr Janet Wiles, University of Queensland
Dr Robert Williamson, Australian National University

Local Committee
Dr Andrew Back, University of Queensland
Dr Phil Diamond, University of Queensland
Prof Joachim Diederich, Queensland University of Technology
Dr Shlomo Geba, Queensland University of Technology
Prof Graeme Halford, University of Queensland
Dr Michael Humphreys, University of Queensland
Dr Brian Lovell, University of Queensland
Mr David Lovell, University of Queensland
Dr Mark Schulz, University of Queensland
Dr Joaquin Sitte, Queensland University of Technology
Dr Guy Smith, University of Queensland
Dr Janet Wiles, University of Queensland
Dr Robert Young, Queensland Dept of Primary Ind

AUTOMATIC GENERATION OF FUZZY RULES BY FUZZY COUNTERPROPAGATION NEURAL NETWORK

H. N. Cheung
Dept. of Computer and Communication Engineering

I Made Wiryana and J.W.L. Millar
Dept. of Computer Science

Edith Cowan University
Joondalup, Western Australia

Abstract

Function approximation is one of the neural-network applications. The counterpropagation network which has fast learning capability tends to produce discontinous function approximation. This paper reports the development of a fuzzy counterpropagation network which has fast learning capability and can perform a continuous function approximation. The proposed network consists of a fuzzy clustering layer, a defuzzifier output node and a modified training procedure. A chaotic time series prediction is used for demonstrating the applicability of the proposed network.

1. Introduction

The Counterpropagation Network (CPN) architecture is simple, fast, and easy to train. CPN has good statistical model representation of the input space. In the network mapping problem, CPN has a closed form of the means square error. The capability of CPN to perform a fast learning is useful for some applications, such as adaptive control, trajectory problem of teach and play robot, adaptive filter, that need an on-line learning mechanism.

Counterpropagation networks can be used for pattern classication where template matching and template interpolation are desired (Hecht-Nielsen, 1988). CPN has been applied in many problems such as Dolphin echolocation (Roitblat et al., 1989), Digital Feedback Equalizer (Manabe and Kaneda, 1991), and it has been implemented in a VLSI system (Kwan and Tsang, 1990).

The CPN architecture is built by combining the Kohonen self-organizing map and the Grossberg Outstar architecture. The Kohonen self-organizing map performs the classification task, and the Groesberg Outstar performs the mapping function. In general, it can be stated that the CPN uses an adaptive table look-up mechanism to perform the mapping while the the table is obtained by training.

To desribe the operation of a typical CPN, let a set of examples of input-output pairs (x_i, y_i), of a function ϕ:

$$\phi : R^n \rightarrow R^m, \; where \; y_i = \phi(x_i) \qquad (1)$$

Assuming that this set of examples cover all the characteristics of the input-output relationship of the function, after training the CPN with this set of examples, the weights of the network are adjusted according to the training set in such a way that the inputs are classifed into clusters. The final clusters can be modelled in the form of a look-up table with N entries, with the number N equal to the number of clusters (Hecht-Nielsen, 1988).

In order for a CPN to perform mapping of continuous functions with good approximation, a large look-up table is required which means that a network with a large middle layer is required (Hecht-Nielsen, 1987). By splitting the middle layer (Lin et al., 1989), the number of neurons in the middle layer can be reduced by a medium amount. As shown by Wang and Mendel (1993), a set of fuzzy rules can be used for a universal function approximation. In our proposed architecture, fuzzy logic is incorporated to the middle layer of a CPN to provide a smoother mapping. Therefore the output of the middle layer gives a membership value of the input belonging to each class. The original Grossberg outstar layer is replaced by a layer of nodes which are controlled by a defuzzifier node.

During training, the modified CPN generates the fuzzy rules from the set of examples of input-output pairs. Each example of input-output pairs is only required to be presented to the input of the network once and no iterative training is required.

In normal operation, this proposed network architecture does not produce a class as a winner in the middle layer, but produces the membership values of the input with respect to each class. By using proper defuzzification method, a continuous function can be obtained at the output layer. Moreover, this mechanism is suitable for on-line learning, because the system does not require iterative learning. Especially, in time series prediction problem, after performing a prediction, the real value of the next time series can be used to refine the network for predicting the following time series.

This method is different from the interpolative mode of conventional CPN. In the interpolative mode, more than one node in the middle layer can win and the winning nodes are weighted with a fraction number and the sum of all weighted number is equal to one. Therefore, there are the first, the second, and the third winner. According to Hecht-Nielsen (1988), in order to perform the interpolative mode, a priori knowledge about the problem is required to define the fraction numbers.

In the proposed network, by the use of fuzzy logic, the number of neurons in the middle layer is greatly reduced which leads to shorter training time, and the network has the capability to perform function approximation without the requirement of a priori knowledge. Similar work in combining fuzzy logic and

the neural system have been reported using different approaches such as Fuzzy ARTMAP (Carpenter et al., 1992), Fuzzy Min-Max (Simpson, 1993).

2. Network Architecture

The proposed network uses fuzzy clustering technique instead of Kohonen layer for the middle layer, and Grossberg Outstar is applied to store the centroid value of each class. By adding a defuzifier node , the defuzzification is based on the centroids obtained in the output layer. The architecture of the network is shown in Fig 2.

W1 is the synaptic weight which stores the information of the prototype of the class (c) and each neuron in middle layer stores the value of farthest deviation of each class (r) and α. W2 is the synaptic weight which represents the information about the output prototype of each class. It will be used in the defuzzification step for calculating the actual output. Therefore an output node is slightly different from a conventional neuron.

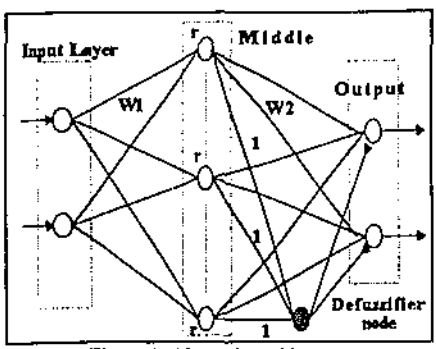

Figure 2. Network architecture

2.1. Classification and Membership Layer

The classification and membership layer, which is the combination of the input and middle layers, partitions the input space into classes. Given an input vector x as shown in Fig. 3, the Euclidean distances between the input vector and all prototype vectors are calculated e.g. d_1, and d_2. By the Euclidean distance, the membership value for each class can be obtained e.g. $m^0(x)$ and $m^1(x)$. In normal operation, the membership value is then used to derive the output vector.

In the training process, the weight between a node in the input layer and that in the middle layer, W1, as shown in Fig. 2, is adjusted to capture the prototype of each class which can be used for the membership value decision. Only weights which connect the closest cluster (highest membership value) with the input vector will be adjusted.

This middle layer works as an IF THEN rule in fuzzy logic term. All of those rules are built up in the training phase automatically and is not pre-defined. This mechanism provides the network a capability to learn the input-output pairs and build the fuzzy rules according to the input-output relationship.

2.2. Defuzzifier node and output layer

The defuzzification process is done by the defuzzifier node and output layer. The defuzzification method used is the centroid method. All weights which connect the defuzzifier node to the middle nodes have values equal to 1. Therefore the output of the defuzzifier node is the total of the membership function of each class. This node is rather different from conventional neurons, due to the adaptive gain characteristic of this node. The gain of this node is controlled by the defuzzifier node.

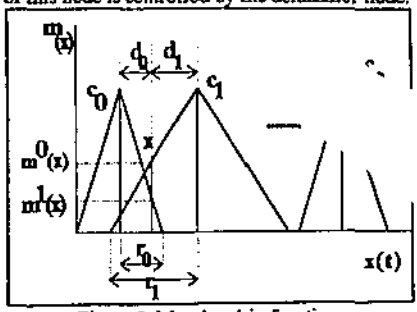

Figure 3. Membership function.

3. Mechanism of the network

3.1. Training mechanism

During training, the learning mechanism partitions the input and output spaces into class and generates the fuzzy rule for each class. The middle node stores the information about the mean of the class and the farthest deviation of the class. The boundary of the class has the membership value equal to zero and the prototype of the class has the membership value equal to one.

Define

r_i the maximum deviation of class i^{th}

c_j the prototype vector of class i^{th} of j dimension

I_j the input vector of j dimension

O_k the target vector of k dimension

The learning mechanism can be formulated as follows :

Step 1. Start with 0 number of clusters

Step 2. New input is applied to the input layer.

Step 3. The Euclidean distance d_i from the input to all clusters that exist are calculated.

$$d_i = \sqrt{\sum_{j=1}^{n} [c_{ji} - I_j]^2} \qquad (2)$$

Step 4. The membership associated with the class that has been created is calculated. For simpler implementation we use the triangular shape for the membership function, u_i. The real number, f_i, determines the shape of membership functions.

$$\mu_i = \left(1 + \left(\frac{d_i}{(r_i - d_i)}\right)^{f_i}\right)^{-1} \qquad (3)$$

Step 5. The weight of the cluster which produces the highest membership value is updated , according to :

$$W1_{win}^{new} = W1_{win}^{old} + \beta\left(I - W1_{win}^{old}\right)$$

$$W2_{win}^{new} = W2_{win}^{old} + \beta\left(I - W2_{win}^{old}\right) \qquad (4)$$

$$\beta = \frac{\alpha}{\alpha+1} \qquad (5)$$

where α is the decay function, which has the initial value equal to 1.0. In this model the value of α for each training can be calculated iteratively by :

$$\alpha_{win}^{new} = \left(\frac{1}{\alpha_{win}^{old}} + 1\right)^{-1} \qquad (6)$$

In this case, the farthest deviation is updated according to :

$$r_{win}^{new} = r_{win}^{old} + \sqrt{\sum_{j=1}^{n}\left[W1_{j,win}^{new} - W1_{j,win}^{old}\right]^2} \qquad (7)$$

Step 6. If the input pattern has zero membership value to all classes, a new class is created and the input pattern is used as the prototype vector of the new class. In this case, the farthest deviation is set to the minimum value. This minimum value determines the resolution of the function approximation.

$$W1^{new} = I \quad ; \quad W2^{new} = O \quad ; r^{new} = \rho \qquad (8)$$

Step 7. If all clusters have been used, choose the closest class by finding the minimal value of D_i:

$$D_i = \left(\sqrt{\sum_{j=1}^{n}\left[W1_{ji} - I_j\right]^2}\right) - r_i \qquad (9)$$

Step 8. Repeat step 2 through 6 for all other training patterns.

3.2. Normal Operation Mechanism

In normal operation, when an input vector is given, the middle layer calculates its membership value to each class using Equation (2) and (3). All the membership values are defuzzified using the centroids from the output layer and the defuzzifier node as :

Output of defuzzifier node is :

$$Out_{DF} = \sum_{i=0}^{p-1} \mu_i \qquad (10)$$

where p is the number of clusters.

Thus the output layer is equal to:

$$Out_k = G\left[\sum_{i=0}^{p-1}\left(\mu_i \; W2_{ik}\right)\right] \qquad (11)$$

where G is the gain factor which corresponds to the output of the defuzzifier node:

$$G = \frac{1}{Out_{DF}} \qquad (12)$$

The output of the k^{th} output node is :

$$Out_k = \frac{\sum_{i=0}^{p-1}\left(\mu_i W2_{ik}\right)}{\sum_{i=0}^{p-1}\mu_i} \qquad (13)$$

4 Simulation Result

A simulation of the proposed model is performed to solve the time series prediction problem. Using some of past data, the prediction of next data will be performed. In this paper, the Mackey-Glass chaotic time series will be used to test the performance of the network.

4.1 Problem description

Chaotic time series is a deterministic and non-linear series. Mackey-Glass chaotic time series is generated from the following delay equation (Wang and Mendel, 1992) :

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \qquad (12)$$

where τ value determines the chaotic behavior of the function. In this simulation, τ equal to 30 is used. We choose 9 for input and 1 for output. This means that we predict the future value by using the 9 past values.

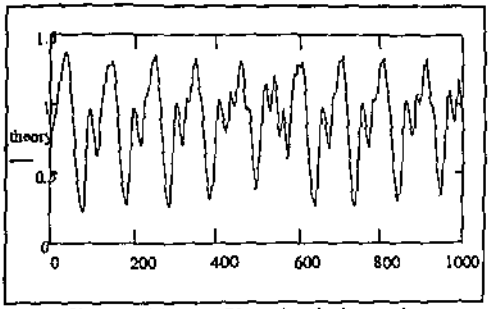

Figure 4. Mackey-Glass chaotic time series

4.2. Simulation result

The simulation is performed by using 80 neurons in the middle layer. In our simulation, the training is performed without repeating the training pattern. To compare the simulation result, the mean square error is used as the performance index.

In the first simulation without on-line adaptation, the first 200 data from 500 to 700 are used as the training patterns. The prediction is performed for the rest of data (701 up to 1000). The result and the comparison with theoretical calculation are shown in Fig 5. The prediction is not too close enough, because the network has not been trained with enough samples and only 60 of the nodes in the middle layer have been used. The mean square error is equal to 0.010715.

The second simulation is performed by using the first 700 data as the training patterns. The result is shown in Fig. 6. By comparing it with theoretical result, it can be shown that the system performs prediction better than the previous experiment. The mean square error is equal to 0.0035749

In Fig 7 and Fig 8, the adaptive on-line training is used. When the on-line training is used for the case that the network has already been trained using the past 700 data, there is no great difference, because the network has converge, and the mean square error is 0.00357490. It shows that the adaptive training does not significantly

affect the performance of the network. When this mechanism is applied to the network which has only been trained from .

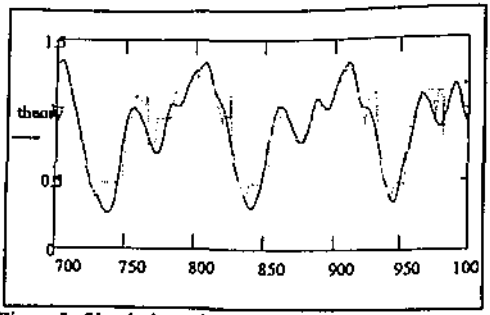

Figure 5. Simulation using x(500) to x(700) as training data

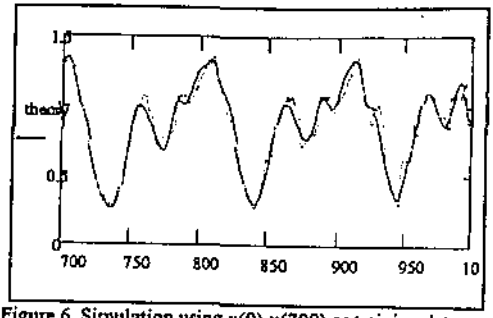

Figure 6. Simulation using x(0)-x(700) as training data

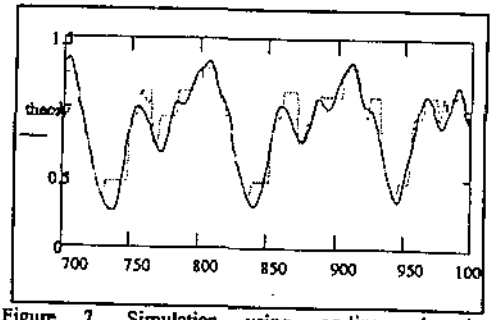

Figure 7. Simulation using on-line adaptation x(500)-x(700) as training pattern

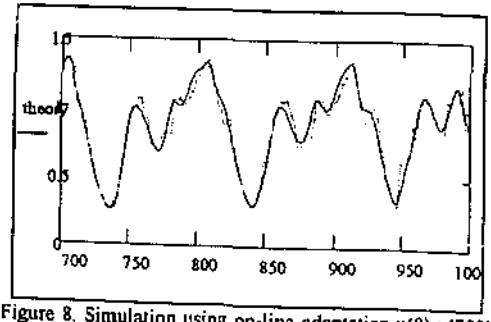

Figure 8. Simulation using on-line adaptation x(0)-x(700) as training pattern

200 data, there is a large improvement. The mean square error is reduced from 0.010715 to 0.009049

5. Conclusion

A new fuzzy feedforward counterpropagation network has been introduced which consists of a fuzzy clustering layer and defuzzifier output node. A modification of the training procedure has been made, and simulation results show that the proposed network architecture perform a continuous function approximation, by learning a set of the input-output pairs. Moreover, the fuzzy rules are generated automatically in the learning process.

The prediction of Mackey-Glass chaotic time series has been demonstrated by applying the proposed network. In addition, it has been shown that the on-line update capability of this network provide the error reduction in the normal operation.

References

Carpenter, Gail A., S Grossberg, N. Markuzon, J. H. Reynolds, D. B. Rosen, (1992), Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps, *IEEE Transactions on Neural Networks*, 3(5), 698- 713.

Hecht-Nielsen, Robert, (1988), Applications of counterpropagation networks, *Neural Networks*, 1, 131-139.

Kwan, Hon-Keung, Pang-Chung Tsang, (1990), Systolic implementation of counterpropataon networks, *ICASSP '90 1990 International Conference on Acoustics, Speech and Signal Processing*, 2, 953-956.

Lin, Z. K. Khorasani, R. V. Patel, (1990), A counterpropagation neural networks for function approximation, *1990 IEEE International Conference on Systems, Man and Cybernetics*, 382-384.

Manabe, Takeshi, Ryuji Kaneda, (1991), Adaptive decision-feedback of digital transmission channels using forward only counterpropagation networks, *1991 IEEE International Joint Conference on Neural Networks*, 1, 220-225.

Roitblat, H. L., P. W. B. Moore, P. E. Nachtigall, R. H. Penner, W. W. L. Au, (1989), Dolphin echolocation: identification of returning echoes using a counterpropagation network, *IJCNN: International Joint Conference on Neural Networks*, 1, 295-300.

Simpson, Patrick K., (1993), Fuzzy min-max neural networks-part2: clustering, *IEEE Transaction on Fuzzy Systems*, 1(1), 32- 45.

Wang, Li-Xin, Jerry M. Mendel, (1992), Generating fuzzy rules by learning from examples, *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6), 1414- 1427.